


CRACKING  
THE CASE WITH

FLEXBOX







Level 1 – Section 1

# Foreshadowing Flexbox

**A Display Type Created for Common Patterns**



# CSS Flexible Box Layout Module Level 1

Flexbox is a collection of CSS properties used to align content and distribute space.

Flex Containers: the `'flex'` and `'inline-flex'` `'display'` values

## Flex Items

Absolutely-Positioned Flex Children  
Flex Item Margins and Paddings  
Flex Item Z-Ordering  
Collapsed Items  
Implied Minimum Size of Flex Items

## Ordering and Orientation

Flex Flow Direction: the `'flex-direction'` property  
Flex Line Wrapping: the `'flex-wrap'` property  
Flex Direction and Wrap: the `'flex-flow'` shorthand  
Display Order: the `'order'` property  
Reordering and Accessibility

## Flex Lines

## Flexibility

The `'flex'` Shorthand  
Basic Values of `'flex'`  
Components of Flexibility  
The `'flex-grow'` property  
The `'flex-shrink'` property  
The `'flex-basis'` property

## Alignment

Aligning with auto margins

## CSS Flexible Box Layout Module Level 1

W3C Candidate Recommendation, 26 May 2016

### This version:

<http://www.w3.org/TR/2016/CR-css-flexbox-1-20160526/>

### Latest published version:

<http://www.w3.org/TR/css-flexbox-1/>

### Editor's Draft:

<https://drafts.csswg.org/css-flexbox/>

### Previous Versions:

<http://www.w3.org/TR/2016/CR-css-flexbox-1-20160301/>  
<http://www.w3.org/TR/2015/WD-css-flexbox-1-20150511/>  
<http://www.w3.org/TR/2014/WD-css-flexbox-1-20140922/>  
<http://www.w3.org/TR/2014/WD-css-flexbox-1-20140322/>  
<http://www.w3.org/TR/2012/CR-css3-flexbox-20120918/>  
<http://www.w3.org/TR/2012/WD-css3-flexbox-20120612/>  
<http://www.w3.org/TR/2012/WD-css3-flexbox-20120322/>  
<http://www.w3.org/TR/2011/WD-css3-flexbox-20111129/>  
<http://www.w3.org/TR/2011/WD-css3-flexbox-20110322/>  
<http://www.w3.org/TR/2009/WD-css3-flexbox-20090723/>

### Feedback:

[www-style@w3.org](mailto:www-style@w3.org) with subject line "[css-flexbox] ..."

### Test Suite:

[http://test.csswg.org/suites/css-flexbox-1\\_dev/nightly-ua/](http://test.csswg.org/suites/css-flexbox-1_dev/nightly-ua/)

### Editors:

[Tab Atkins Jr.](#) (Google)

- Two new display values
- 11 new properties
- <http://go.codeschool.com/flexbox-spec>
- <http://go.codeschool.com/caniuse-flexbox>

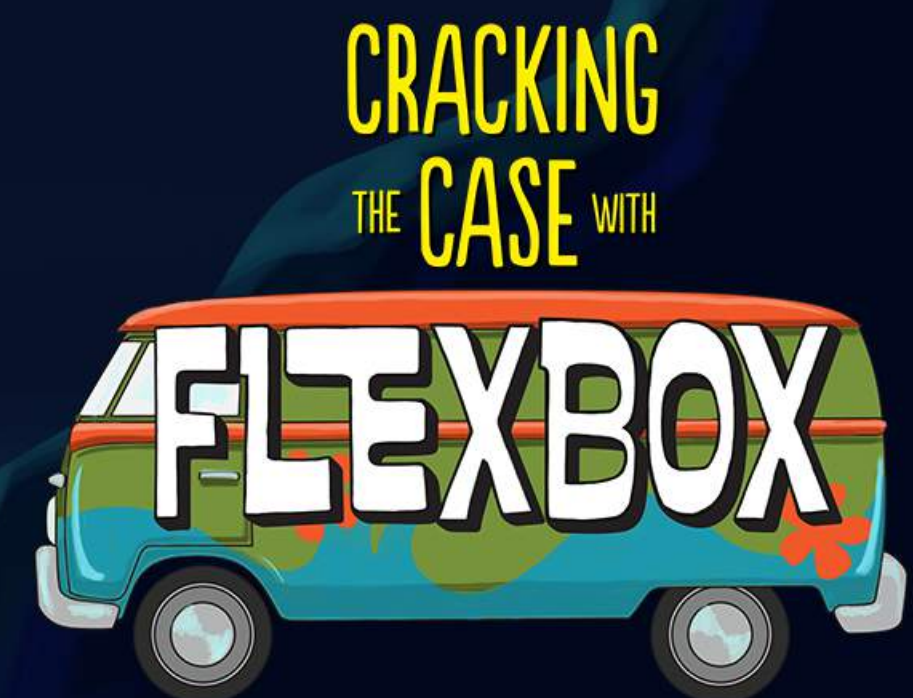
CRACKING  
THE CASE WITH





# UI Patterns in Flexbox

- Equal heights
- Vertically centered content
- Media objects
- Sticky footers
- Column layouts





# Flexbox Is a New Display Type

Flexbox adds flex containers, flex items, and flex lines in addition to new properties.

- Block layout
- Table layout
- Inline layout
- Positioning layout
- Flex layout
  - Flex containers
  - Flex items
  - Flex lines

*Learn about other layouts  
and core HTML and CSS in  
these courses.*





# Default Behaviors of Elements

index.html

HTML

```
<article class="setup">This ...  
  <img height="150" height="150" ...>  
  <p>...</p>  
</article>
```

site.css

CSS

```
article {  
  border: 1px solid #d1eda6; }
```

*display: block*



*display: inline*

This pipe was found outside the door of the amusement park office, soon after the ghost was seen flying by.

*display: block*

# Overriding Default Behaviors

index.html

HTML

```
<article class="setup">This ...  
  <img height="150" height="150" ...>  
  <p>...</p>  
</article>
```

site.css

CSS

```
article {  
  border: 1px solid #d1eda6;  
  display: inline; }
```

*Collapsed*



This pipe was found outside the door of the amusement park office, soon after the ghost was seen flying by.



# Flex Containers Control Layout of Child Items

index.html

HTML

```
<article class="setup">This ...  
  <img height="150" height="150" ...>  
  <p>...</p>  
</article>
```

site.css

CSS

```
article {  
  border: 1px solid #d1eda6;  
  display: flex; }
```

*Flex container*



This pipe was found outside the door of the amusement park office, soon after the ghost was seen flying by.

*Aligned horizontally*



# display: flex Creates a Block-level Container

index.html

HTML

```
<article class="setup">This ...  
  <img height="150" height="150" ...>  
  <p>...</p>  
</article>
```

site.css

CSS

```
article {  
  border: 1px solid #d1eda6;  
  display: flex; }
```



This pipe was found outside the door of the amusement park office, soon after the ghost was seen flying by.

*Uses all available width*



# display: inline-flex Creates an Inline Container

index.html

HTML

```
<article class="setup">This ...  
  <img height="150" height="150" ...>  
  <p>...</p>  
</article>
```

site.css

CSS

```
article {  
  border: 1px solid #d1eda6;  
  display: inline-flex; }
```



This pipe was found outside the door of the amusement park office, soon after the ghost was seen flying by.

*Container adjusts to  
width of the content*



# Flex Items Are Distributed Along Flex Lines

Flex items are direct children of a flex container and can be HTML elements or text.

index.html

HTML

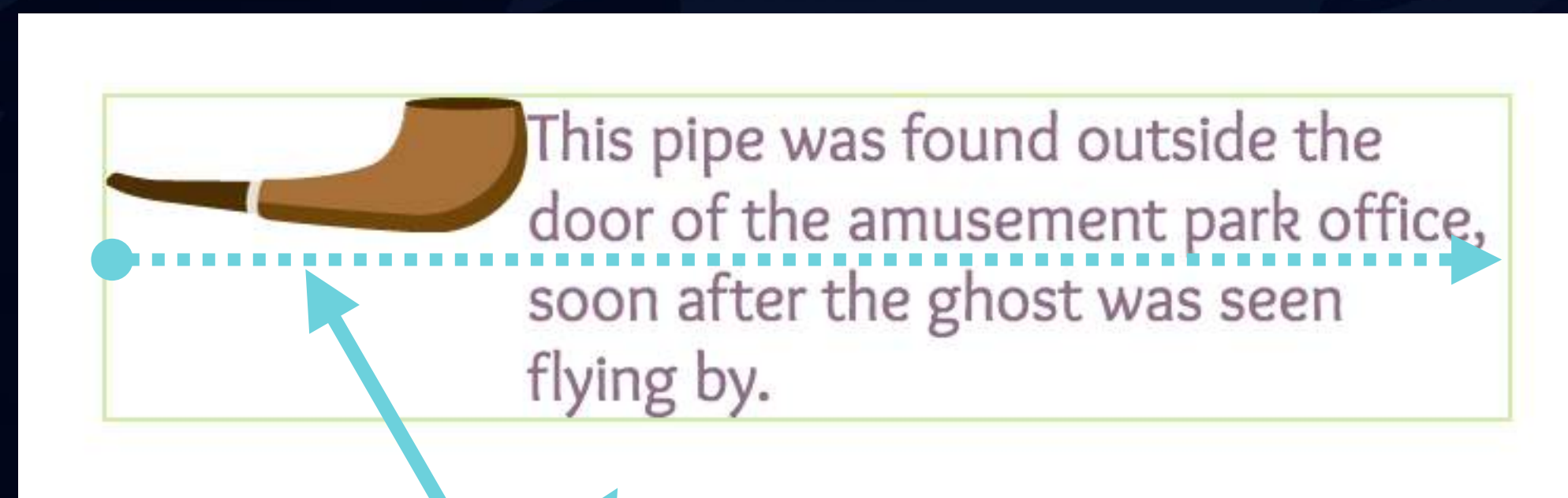
```
<article class="setup">This ...  
  <img height="150" height="150" ...>  
  This pipe was found outside...  
</article>
```

site.css

CSS

```
article {  
  border: 1px solid #d1eda6;  
  display: flex; }
```

*No HTML element — just text*



*Two flex items fill the available space of a horizontal flex line*



# Flex Item Spacing

A spacing algorithm is used to distribute space of all flex items across a single flex line.

index.html

HTML

```
<article class="setup">This ...  
  <img height="150" height="150" ...>  
  <h3>...</h3>  
  <p>...</p>  
</article>
```

site.css

CSS

```
article {  
  border: 1px solid #d1eda6;  
  display: flex; }
```

*Three flex items*



Tobacco  
Pipe

This pipe was found  
outside the door of the  
amusement park office,  
soon after the ghost  
was seen flying by.



# Grouping Elements Into Flex Items

Grandchild elements do not become flex items.

index.html

HTML

```
<article class="setup">This ...  
  <img height="150" height="150" ...>  
  <div>  
    <h3>...</h3>  
    <p>...</p>  
  </div>  
</article>
```

site.css

CSS

```
article {  
  border: 1px solid #d1eda6;  
  display: flex; }
```

*Two flex items*

*Two flex items*



## Tobacco Pipe

This pipe was found outside the door of the amusement park office, soon after the ghost was seen flying by.




CRACKING  
THE CASE WITH

FLEXBOX







Level 1 – Section 2

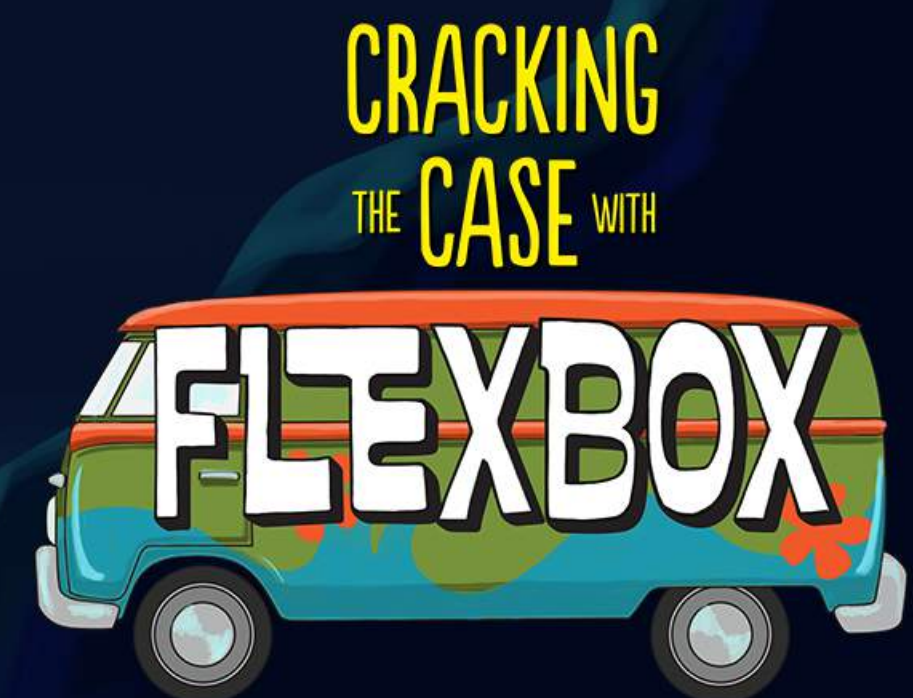
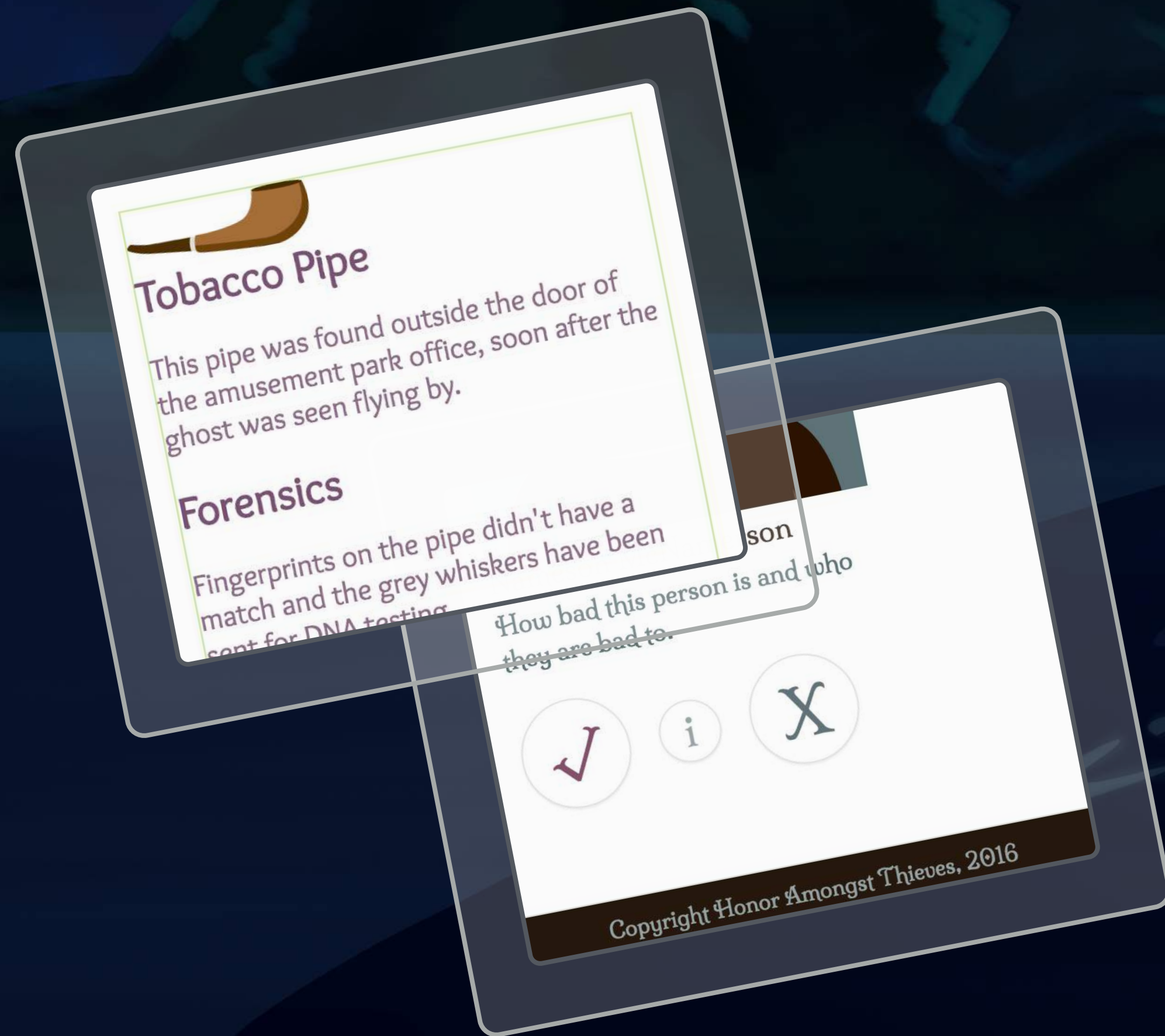
# Foreshadowing Flexbox

Wily Wrapping



# Flexbox Wrapping and Direction Examples

- Displaying flex items across multiple lines
- Changing the direction of flex lines
- Distributing space vertically





# Problem: Wanting Flex Items to Stack

index.html

HTML

```
<article class="clue">
  <img height="150" height="150" ...>
  <div>
    <h3>...</h3>
    <p>...</p>
    <h3>...</h3>
    <p>...</p>
  </div>
</article>
```

*These could go below the image*

*320px wide*



## Tobacco Pipe

This pipe was found outside the door of the amusement park office, soon after the ghost was seen flying by.

## Forensics

Fingerprints on the pipe didn't have a match and the grey whiskers have been sent for DNA testing.



# Solution: flex-wrap

This property's default value is nowrap and it accepts: nowrap | wrap | wrap-reverse.

index.html

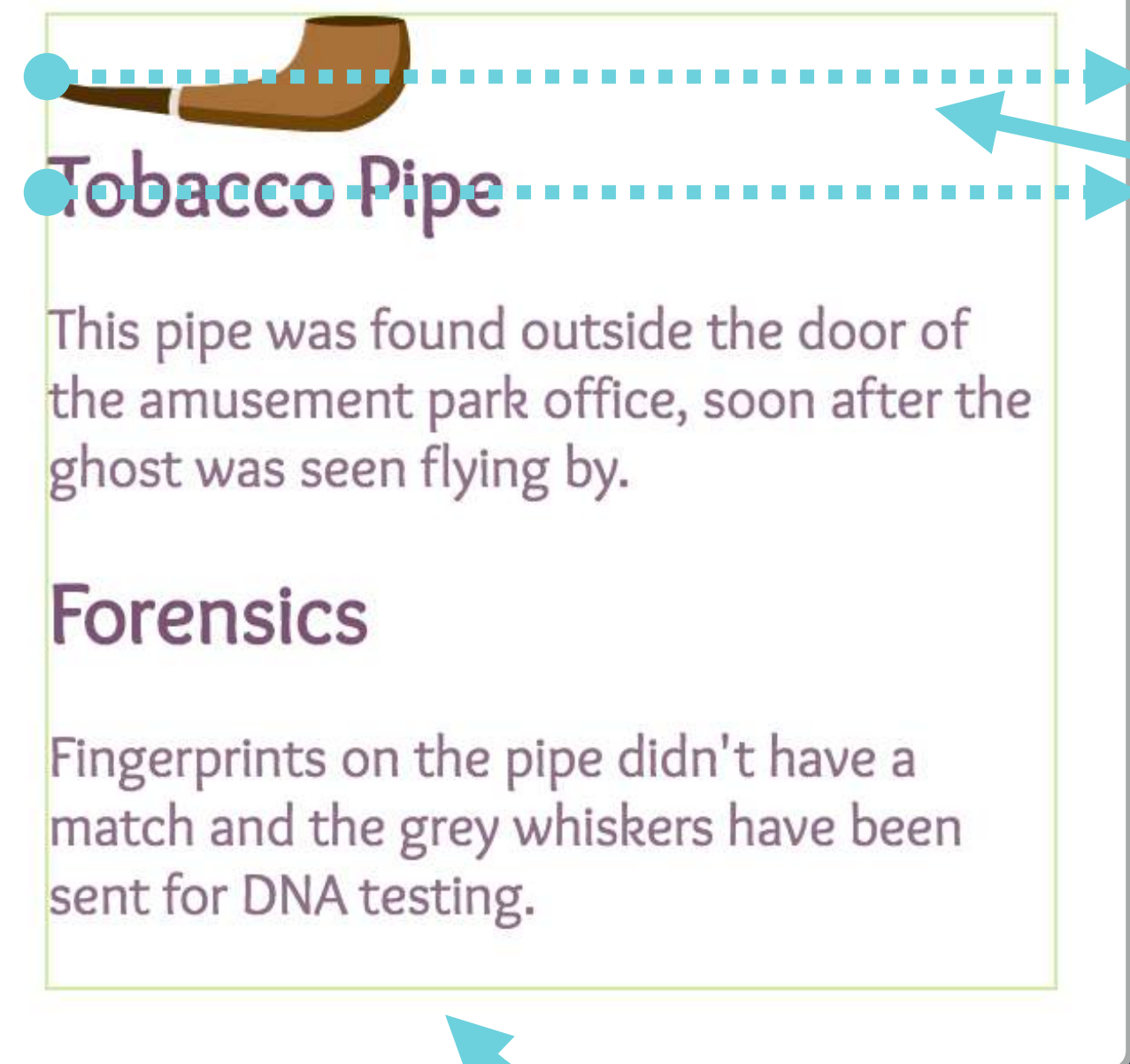
HTML

```
<article class="clue">...</article>
```

site.css

CSS

```
.clue {  
  border: 1px solid #d1eda6;  
  display: flex;  
  flex-wrap: wrap;  
}
```



The div can't fit here, so it wraps

320px wide



# Put It in Reverse?

The first of the flexbox properties that allows for rendering items in reverse.

index.html

HTML

```
<article class="clue">...</article>
```

site.css

CSS

```
.clue {  
  border: 1px solid #d1eda6;  
  display: flex;  
  flex-wrap: wrap-reverse; }  
}
```

## Tobacco Pipe

This pipe was found outside the door of the amusement park office, soon after the ghost was seen flying by.

## Forensics

Fingerprints on the pipe didn't have a match and the grey whiskers have been sent for DNA testing.



*The flex items reverse in order*



# Problem: Wanting Flex Items to Always Stack

index.html

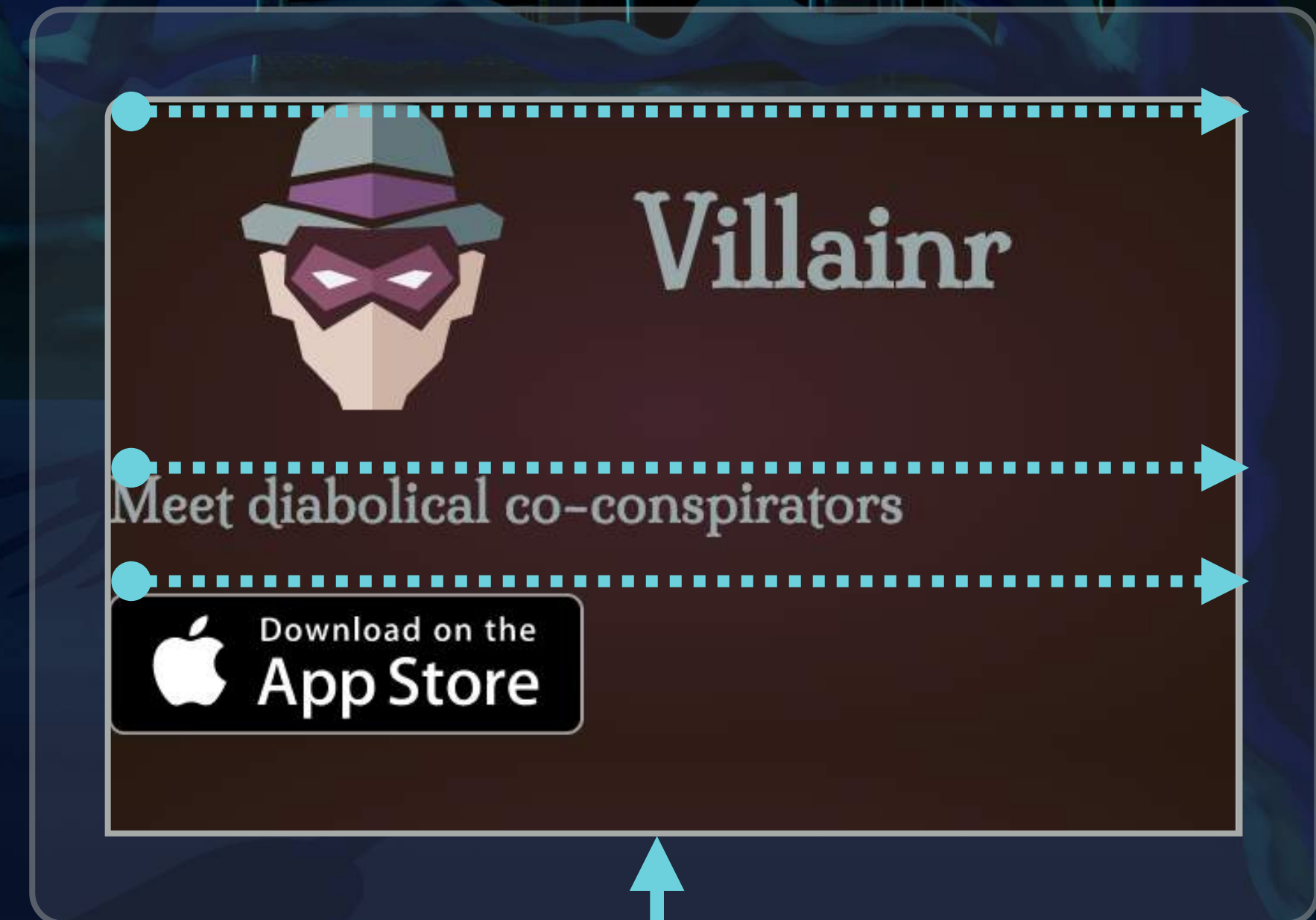
HTML

```
<main>
  <img ...>
  <h1>...</h1>
  <p>...</p>
  <a>...</a>
</main>
```

site.css

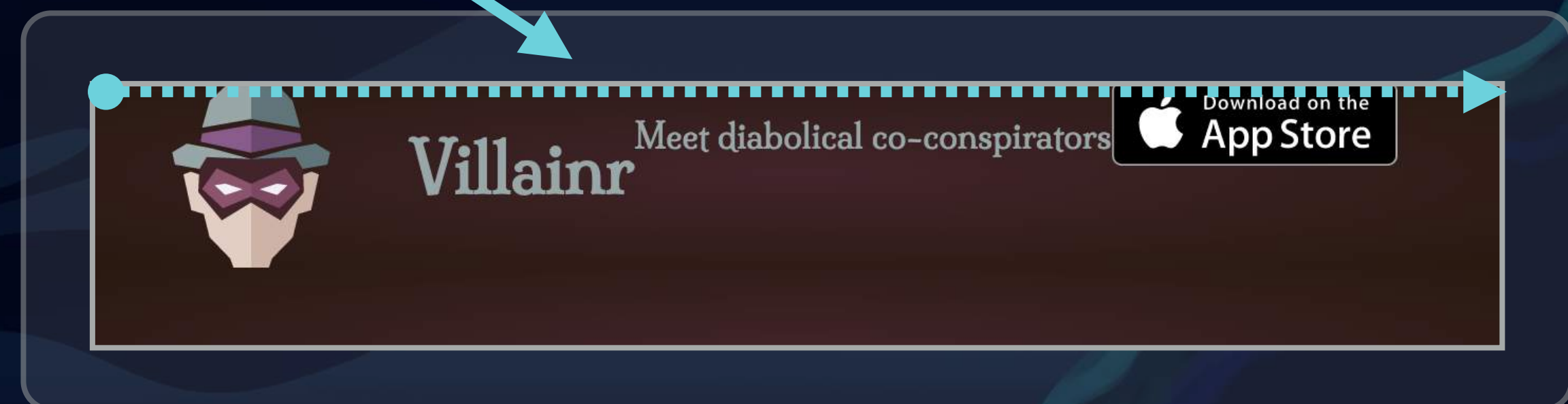
CSS

```
main {
  display: flex;
  flex-wrap: wrap; }
```



667px wide

320px wide





# Solution: flex-direction

The default value is `row` and it accepts: `row` | `row-reverse` | `column` | `column-reverse`.

index.html

HTML

```
<main>
  <img ...>
  <h1>...</h1>
  <p>...</p>
  <a>...</a>
</main>
```

site.css

CSS

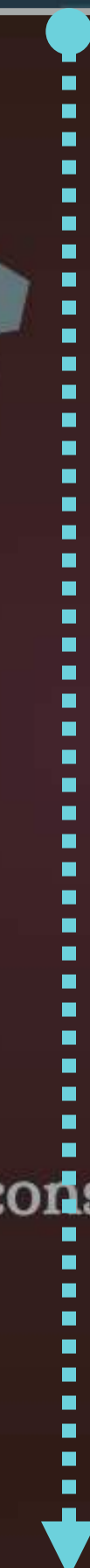
```
main {
  display: flex;
  flex-direction: column; }
```

*Flex items  
display  
vertically*



Villainr

Meet diabolical co-conspirators





# Page Layout With flex-direction

Main elements for layout are flex items with the `column` direction.

index.html

HTML

```
<body>
  <header>...</header>
  <main>...</main>
  <footer>...</footer>
</body>
```

site.css

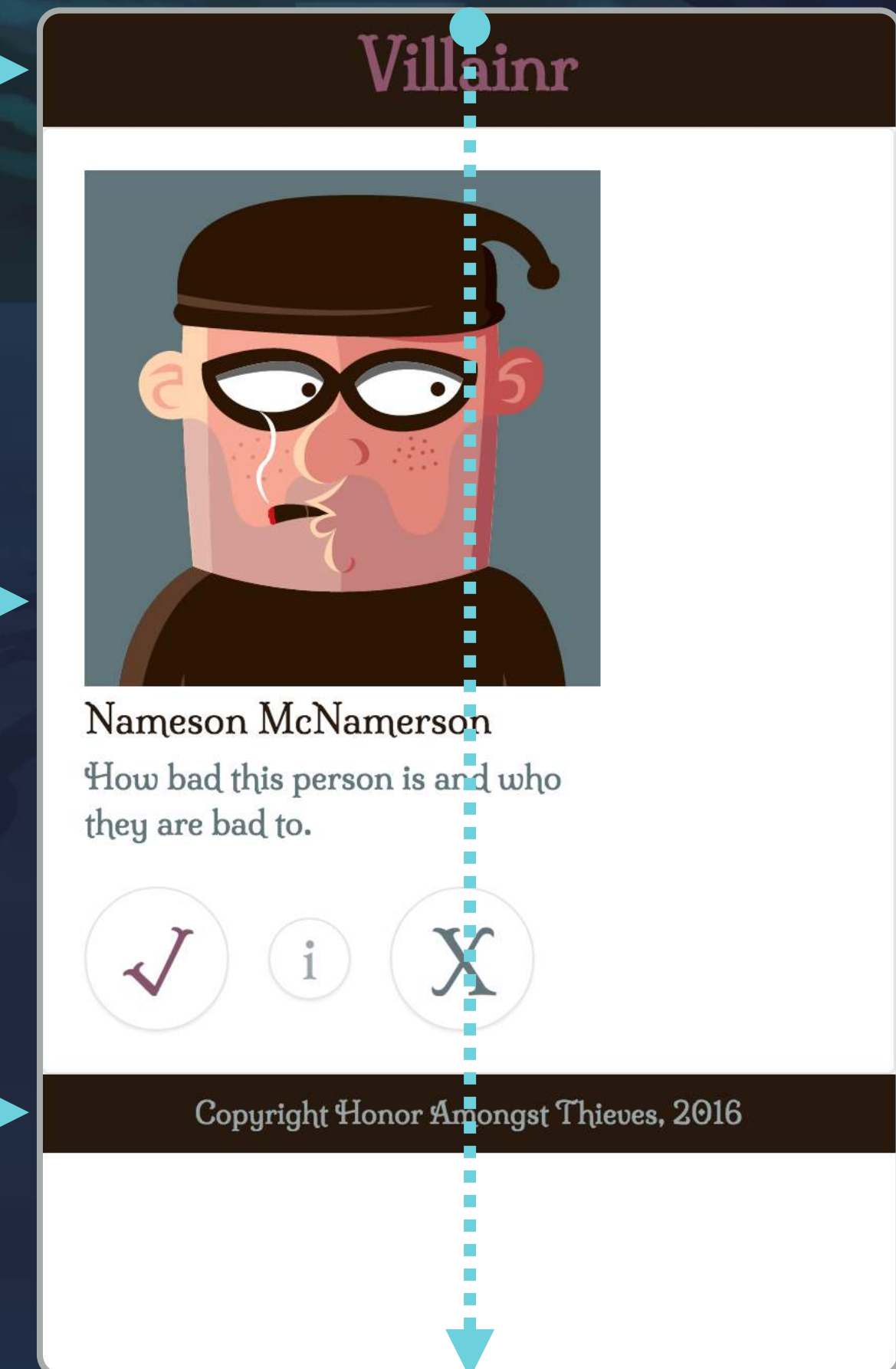
CSS

```
body {
  ...
  display: flex;
  flex-direction: column; }
```

header

main

footer





# Mixing in Some Non-flexbox CSS

Setting the page elements to 100% height uses all available space.

index.html

HTML

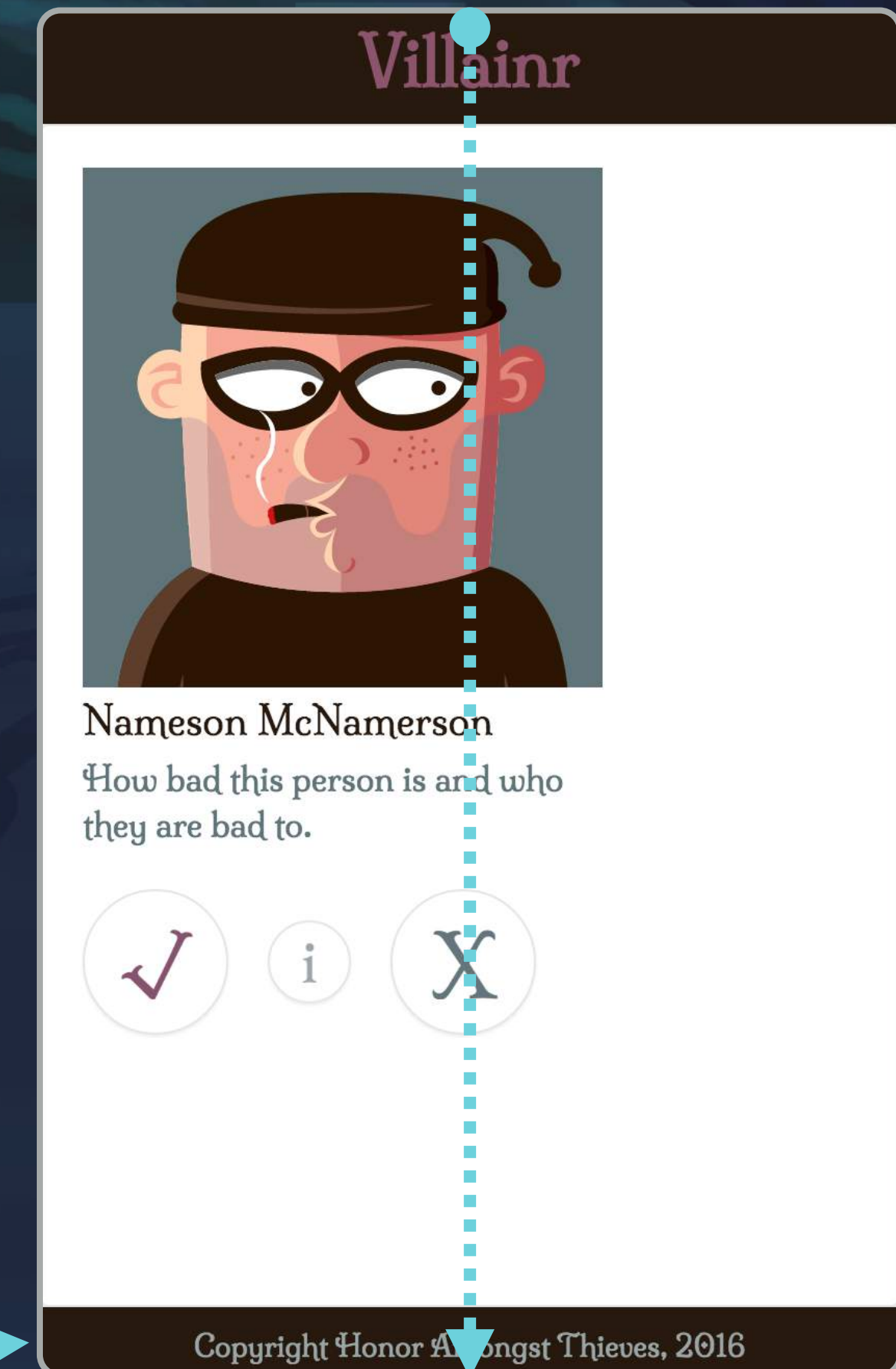
```
<body>
  <header>...</header>
  <main>...</main>
  <footer>...</footer>
</body>
```

site.css

CSS

```
html, body, main {
  height: 100%; }
```

*Stays at the bottom  
of the screen*





CRACKING  
THE CASE WITH

FLEXBOX







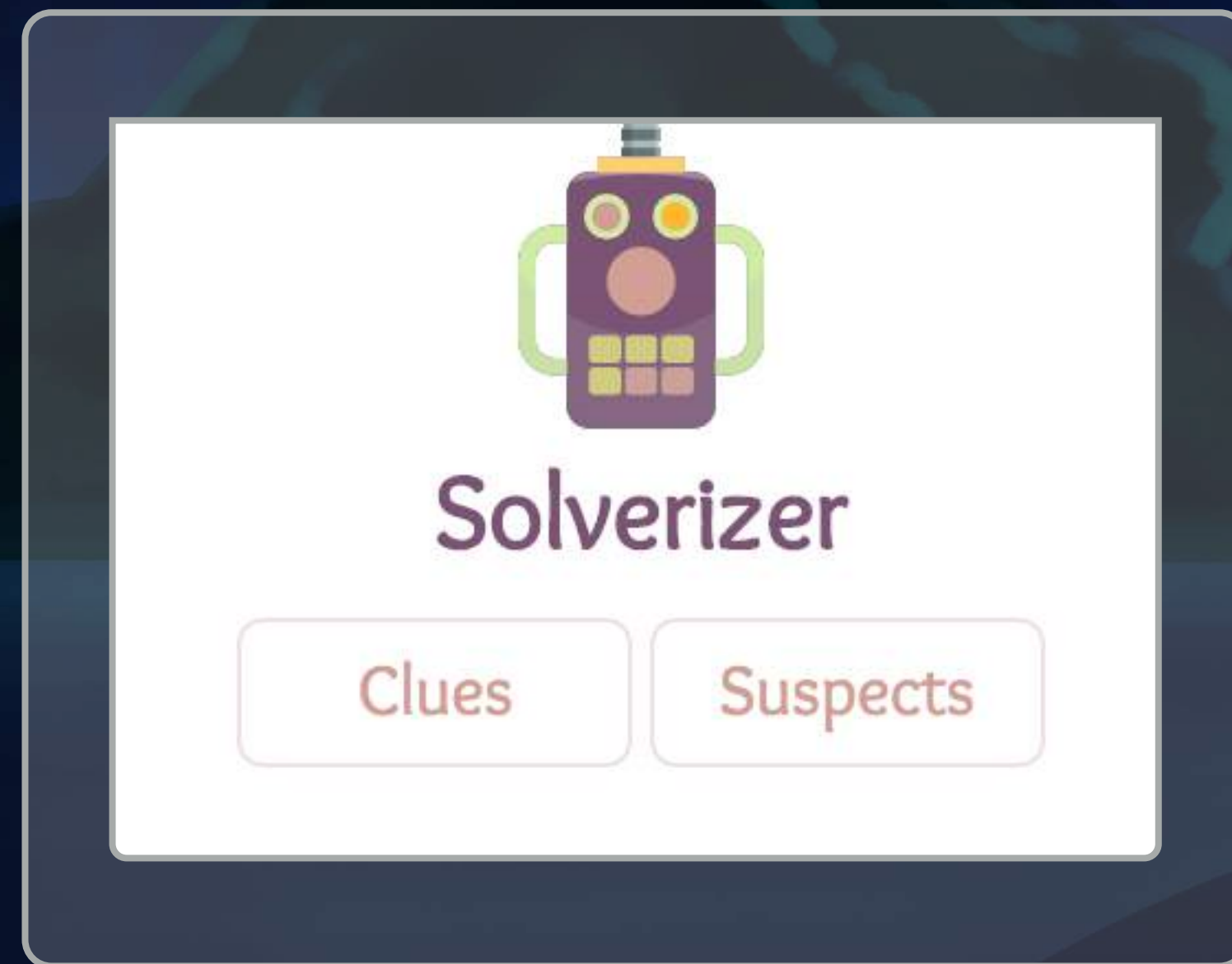
Level 2 – Section 1

# Justification and Order

Distributing Space Along Flex Lines

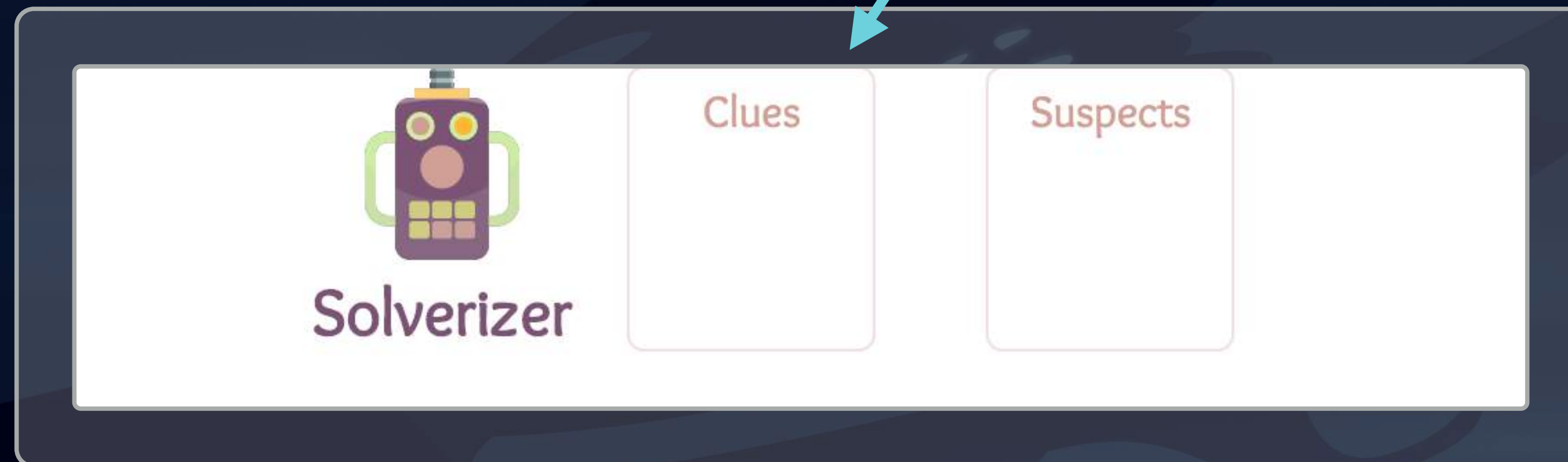


# Example Patterns



*Positioning elements vertically*

*Positioning elements horizontally*





# Flex Items Are Distributed Along Axes

The main axis is determined by the `flex-direction`.

index.html

HTML

```
<nav>
  <a ...>...</a>
  <a ...>...</a>
  <a ...>...</a>
</nav>
```

site.css

CSS

```
@media screen and (min-width: 375px) {
  nav {
    display: flex; } }
```

*start*

*center*

*end*



Solverizer

Clues

Suspects



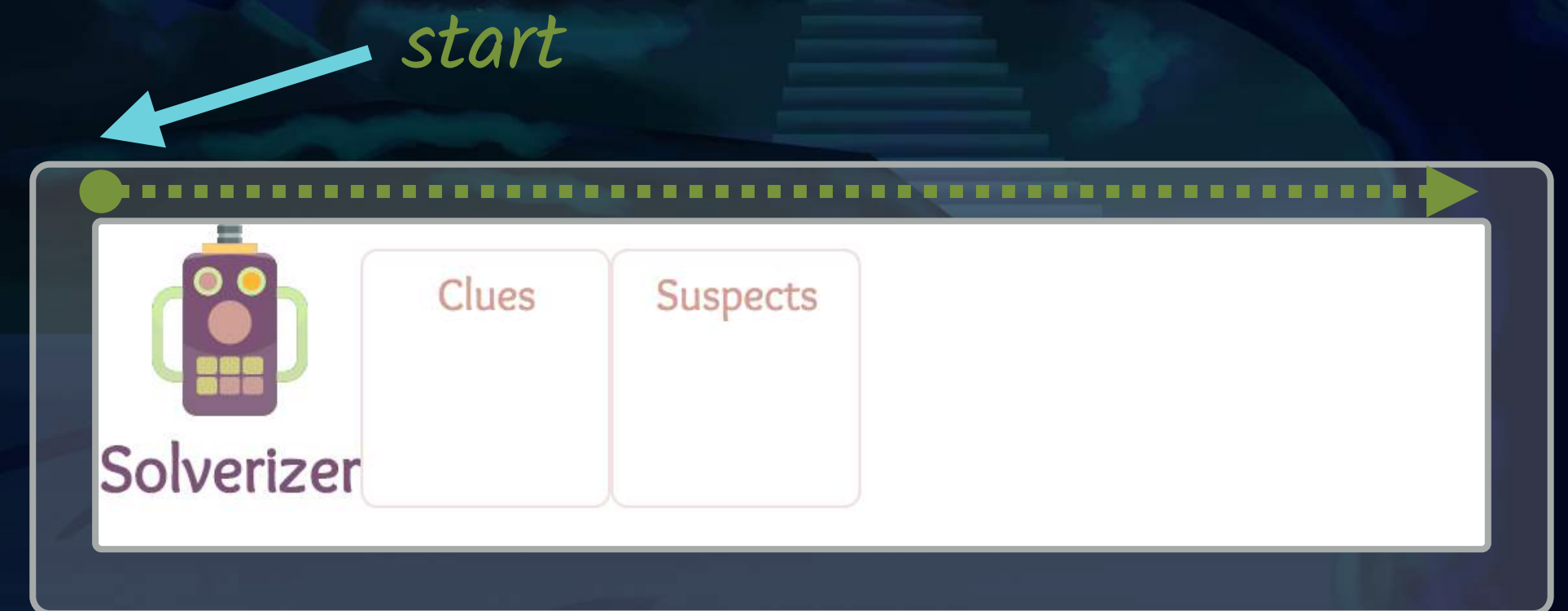
# justify-content

Used to distribute space on the main axis, the default value is **flex-start** and it accepts:  
**flex-start** | **flex-end** | **center** | **space-between** | **space-around**.

site.css

CSS

```
@media screen and (min-width: 375px) {  
  nav {  
    display: flex;  
    justify-content: flex-start; } }
```



site.css

CSS

```
@media screen and (min-width: 375px) {  
  nav {  
    display: flex;  
    justify-content: flex-end; } }
```





# Spacing Values for justify-content

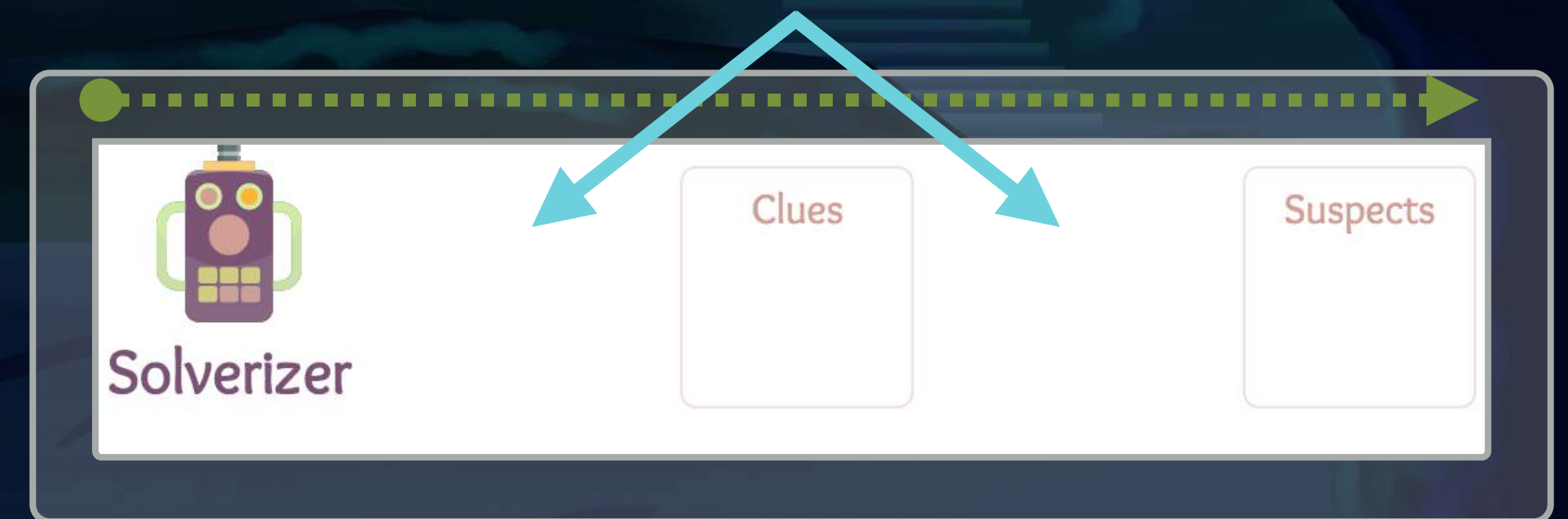
Flexbox's spacing algorithm distributes available space between elements.

site.css

CSS

```
@media screen and (min-width: 375px) {  
  nav {  
    display: flex;  
    justify-content: space-between; } }
```

*Flush to edges, space in between*



site.css

CSS

```
@media screen and (min-width: 375px) {  
  nav {  
    display: flex;  
    justify-content: space-around; } }
```

*Space added around each element*





# Our Example Uses center

index.html

HTML

```
<nav>
  <a ...>...</a>
  <a ...>...</a>
  <a ...>...</a>
</nav>
```

site.css

CSS

```
@media screen and (min-width: 375px) {
  nav {
    display: flex;
    justify-content: center; } }
```

*Content center aligns with container center*





# Adding Space With Non-flexbox CSS

Margin and padding can be applied to flex items.

index.html

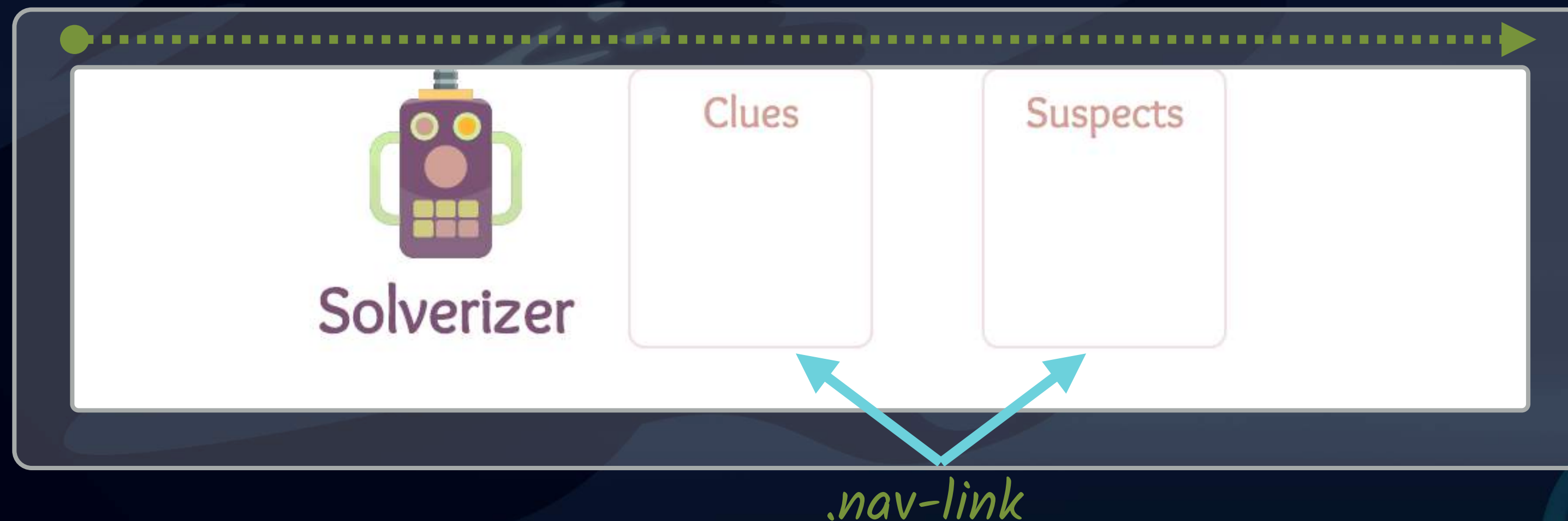
HTML

```
<nav>
  <a class="nav-brand">...</a>
  <a class="nav-link">...</a>
  <a class="nav-link">...</a>
</nav>
```

site.css

CSS

```
@media screen and (min-width: 375px) {
  nav {
    display: flex;
    justify-content: center; }
  .nav-link {
    margin: 0 20px;} }
```





# Justifying Content in a Column

Main elements for layout are flex items with the `column` direction.

index.html

HTML

```
<main>
  ...
</main>
```

site.css

CSS

```
main {
  display: flex;
  flex-direction: column;
  justify-content: center; }
```

*start*

*center of column*

*end*






CRACKING  
THE CASE WITH

FLEXBOX







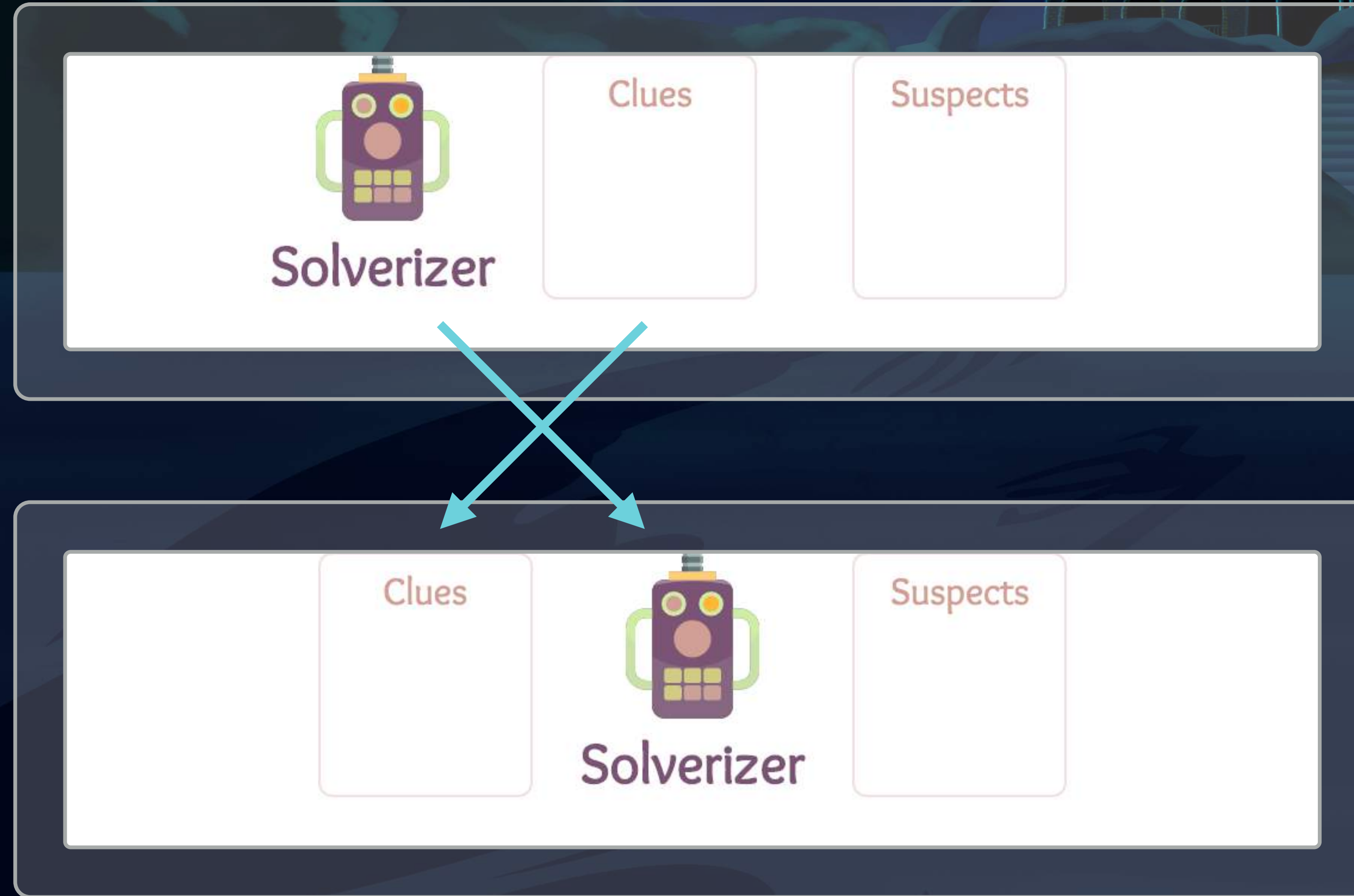
Level 2 – Section 2

# Justification and Order

Changing the Order of Items With CSS



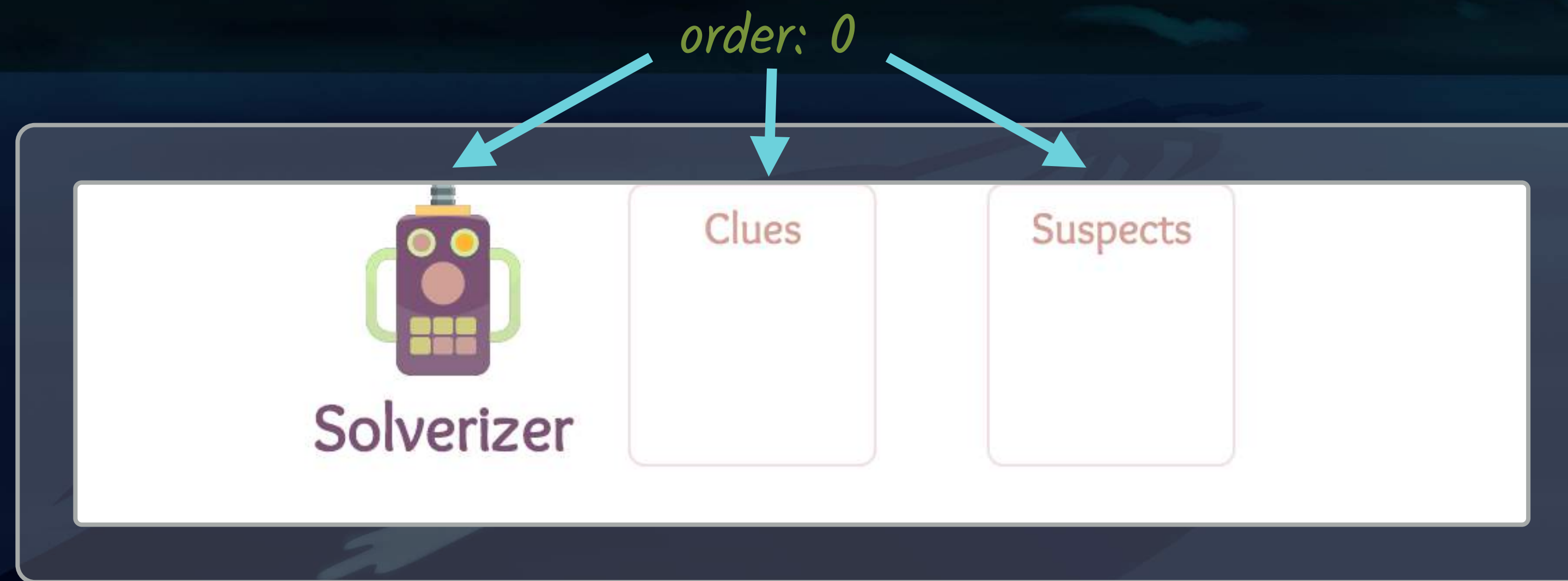
# Display Position Can Be Changed With CSS





# order

**order** is used to determine the order of flex items along the main axis. It defaults to 0 and accepts positive and negative numbers.





# Changing the Order of a Flex Item

Using a negative number displays an item first, based on the `flex-direction`.

index.html

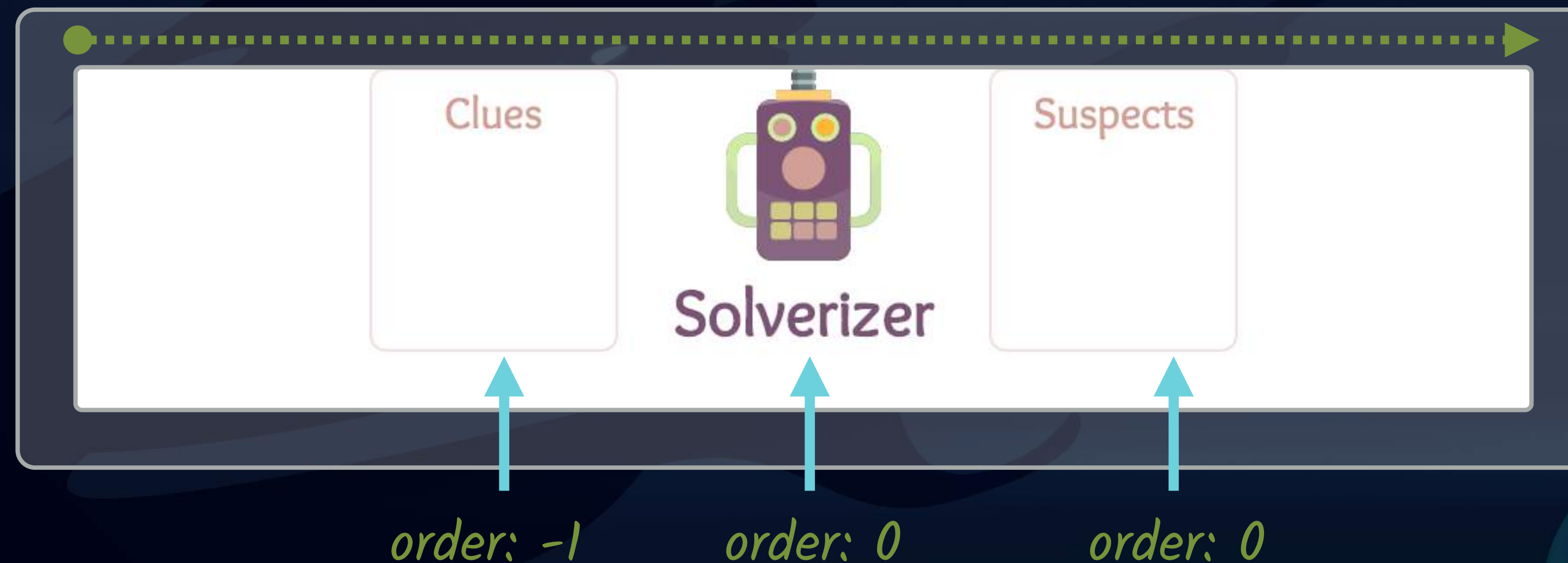
HTML

```
<nav>
  <a class="nav-brand">...
  <a class="nav-clues">...
  <a class="nav-suspects">...
</nav>
```

site.css

CSS

```
@media screen and (min-width: 375px) {
  ...
  .nav-clues {
    order: -1; } }
```





# Changing Order of a Flex Item in a Column

Order changes the position of items along the main axis of the column.

index.html

HTML

```
<main>
  <img ...>
  <h1>...</h1>
  <p class="slogan">...</p>
  <a>...</a>
</main>
```

site.css

CSS

```
...
.slogan {
  order: 1;
  margin-top: 10px; }
```

order: 0

order: 0

order: 0

order: 1



Villainr



Meet diabolical co-conspirators



CRACKING  
THE CASE WITH

FLEXBOX







Level 3

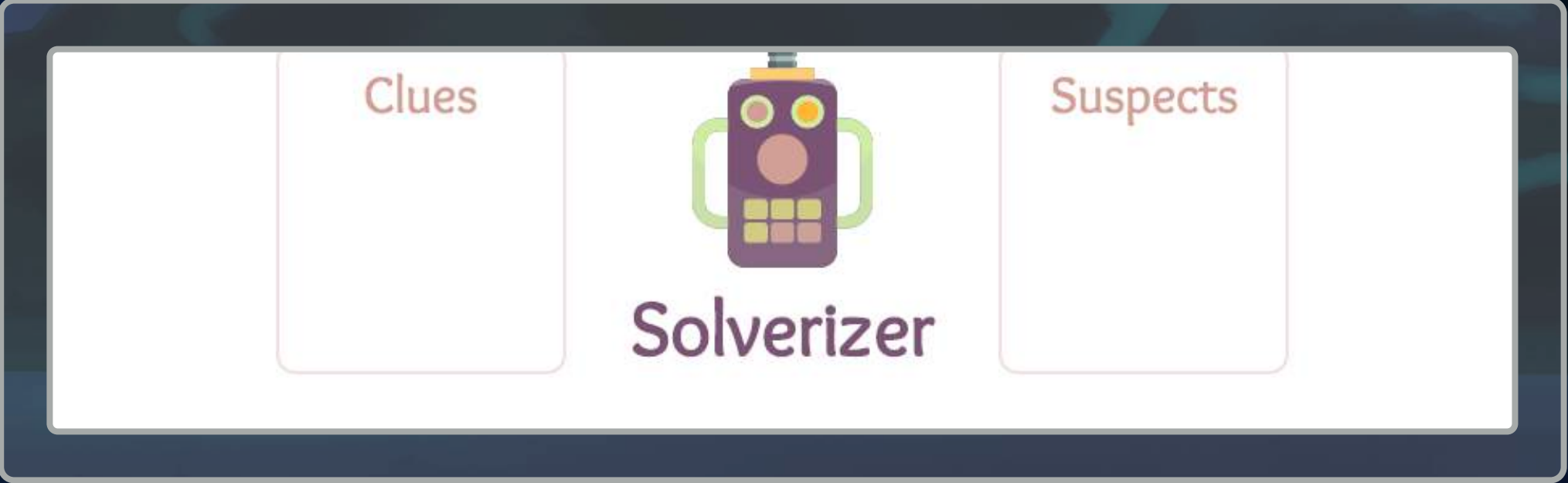
# Aligning Alibis

**Distributing Space Between Items**



# Distributing Space on the Cross Axis

Before



After



Before



After





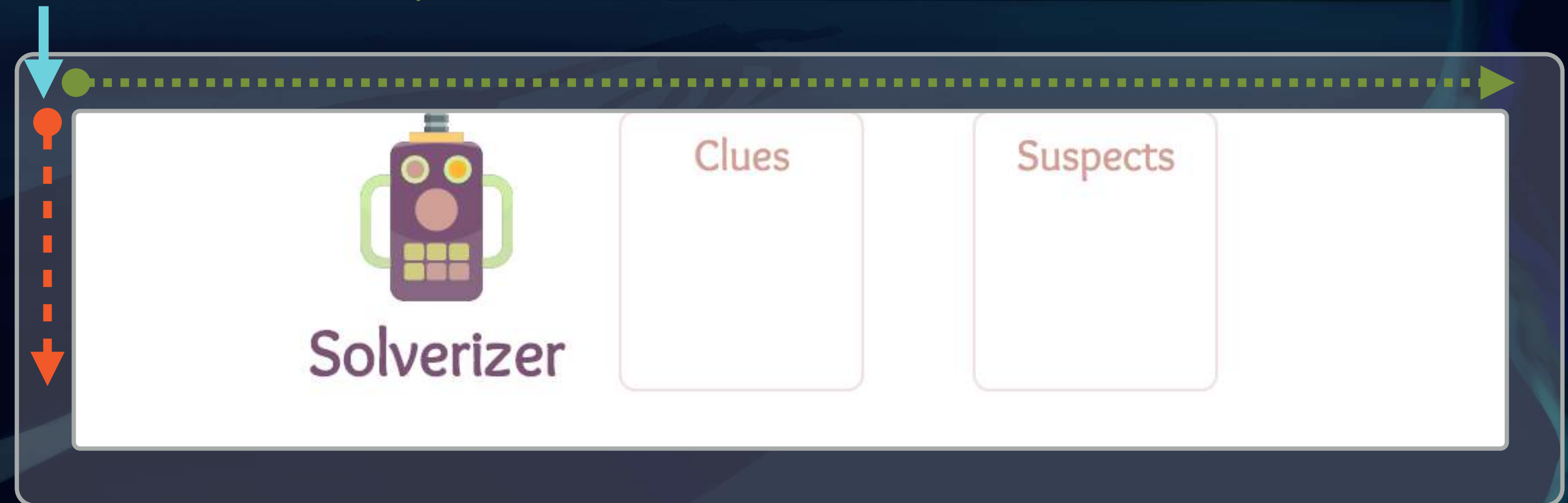
# Flex Items Are Aligned Along a Cross Axis

The cross axis is perpendicular to the main axis.



*In column direction, the cross axis is horizontal*

*In row direction, the cross axis is vertical*





# align-items

Used to align content on the cross axis, the default value is **stretch** and it accepts:  
**stretch** | **flex-start** | **flex-end** | **center** | **baseline**.

index.html

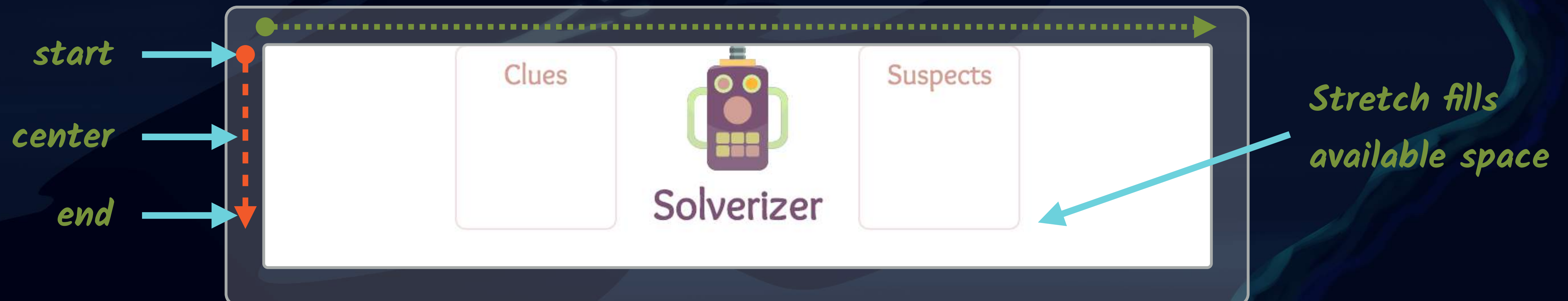
HTML

```
<nav>
  <a class="nav-brand">...
  <a class="nav-clues">...
  <a class="nav-suspects">...
</nav>
```

site.css

CSS

```
@media screen and (min-width: 375px) {
  .nav{
    align-items: stretch;
    display: flex;
    justify-content: center; } }
```





# Start and End

site.css

CSS

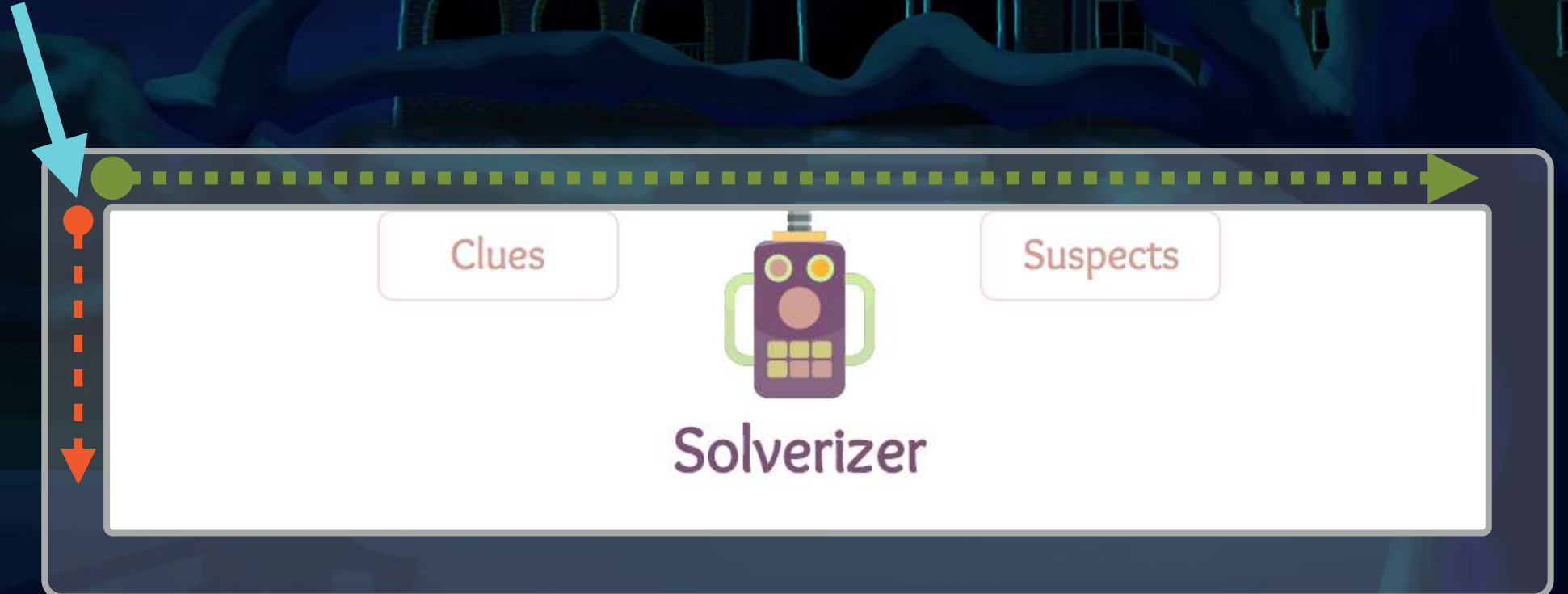
```
@media screen and (min-width: 375px) {  
  .nav{  
    align-items: flex-start;  
    ... } }
```

site.css

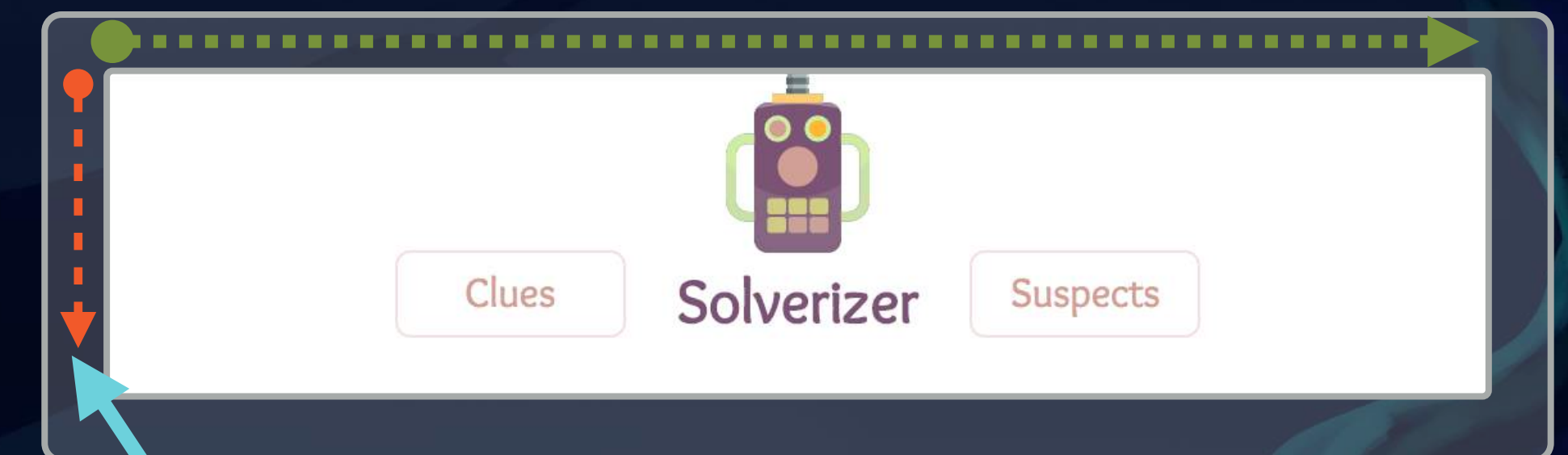
CSS

```
@media screen and (min-width: 375px) {  
  .nav{  
    align-items: flex-end;  
    ... } }
```

start



end





# Baseline and Center

site.css

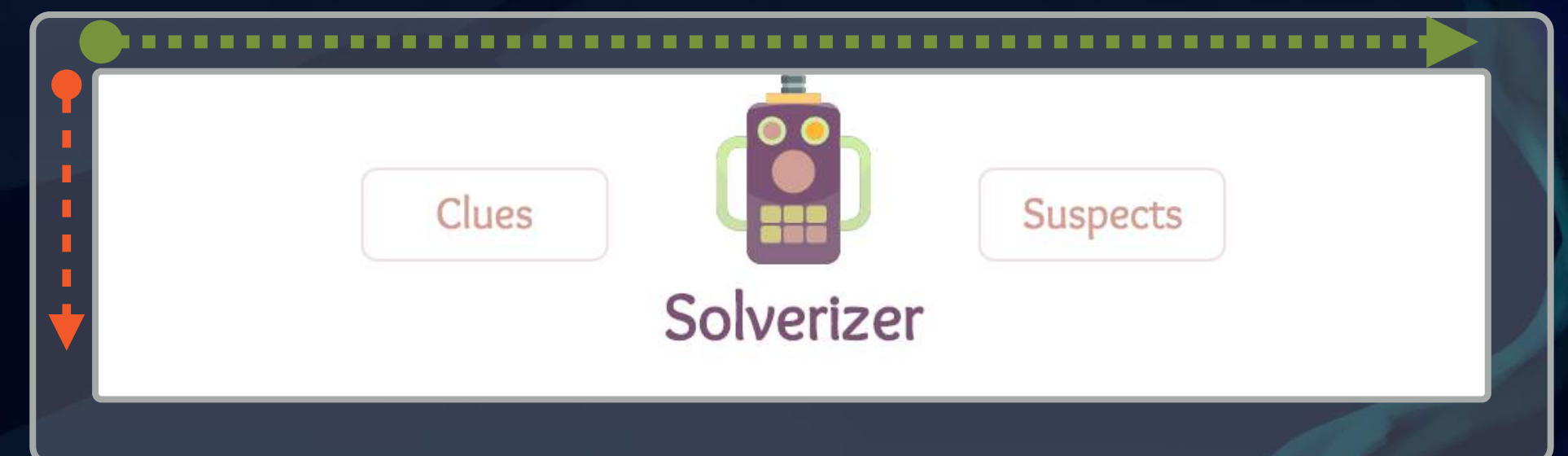
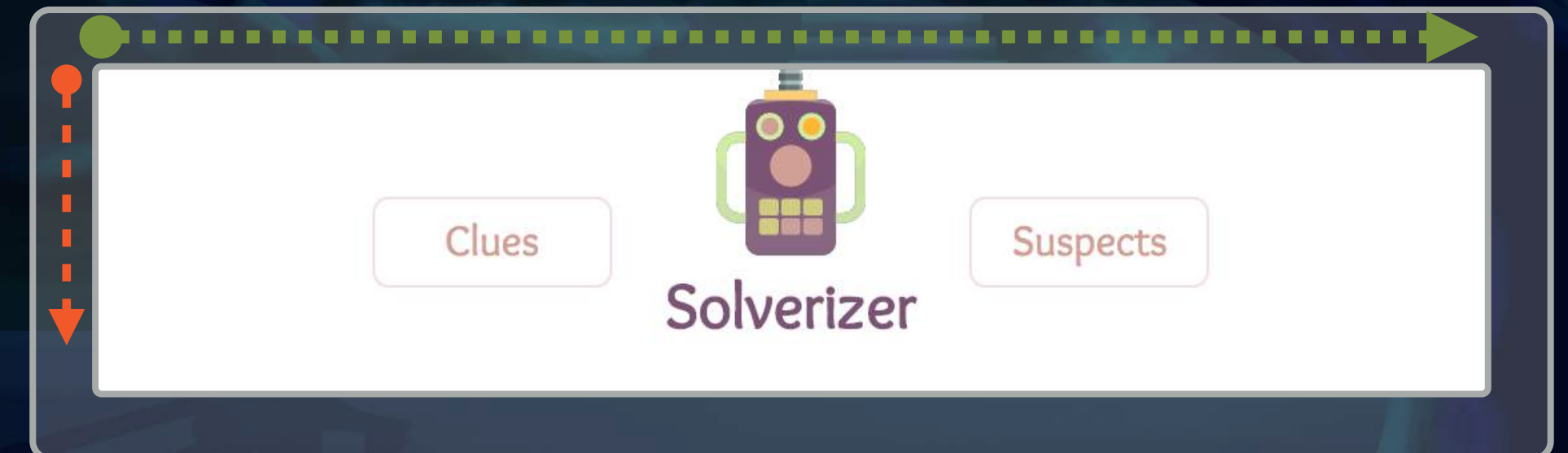
CSS

```
@media screen and (min-width: 375px) {  
  .nav{  
    align-items: baseline;  
    ... } }
```

site.css

CSS

```
@media screen and (min-width: 375px) {  
  .nav{  
    align-items: center;  
    ... } }
```





# Aligning Items in Column Direction

Items stretch to fill the width of the container.

index.html

HTML

```
<main>
  ...
</main>
```

site.css

CSS

```
...
main {
  align-items: center;
  ...
}
```

*Centered horizontally*






CRACKING  
THE CASE WITH

FLEXBOX







Level 4 – Section 1

# Sizing Up the Properties

**New Properties for Space-based Size**



# Examples, Sized With Flex Properties

### Stay In the Know

Sign up below to receive updates when new clues are added.

Email

Send





*Grow to use available width*

### Stay In the Know

Sign up below to receive updates when new clues are added.

Email

Send

	<h4>Brown necktie</h4> <p>The neutral tones help its owner blend in with any background — especially the faded wallpaper in the aging haunted house.</p>		<h4>Tobacco Pipe</h4> <p>The musty smell of old tobacco wafts from this vintage pipe — which wasn't actually vintage when the owner purchased it new.</p>
	<h4>Gold Pocket Watch</h4> <p>The telltale ticking of this pocket watch echoes throughout the halls of the haunted house, bouncing off sheet-covered furniture... or ghosts.</p>		<h4>Cane</h4> <p>There's a dusting of hard-candy sugar along this solid-oak cane, which is ideal for enthusiastic pointing while lecturing whippersnappers.</p>

*Defined widths for items*



# (Mostly) Default Behavior

index.html

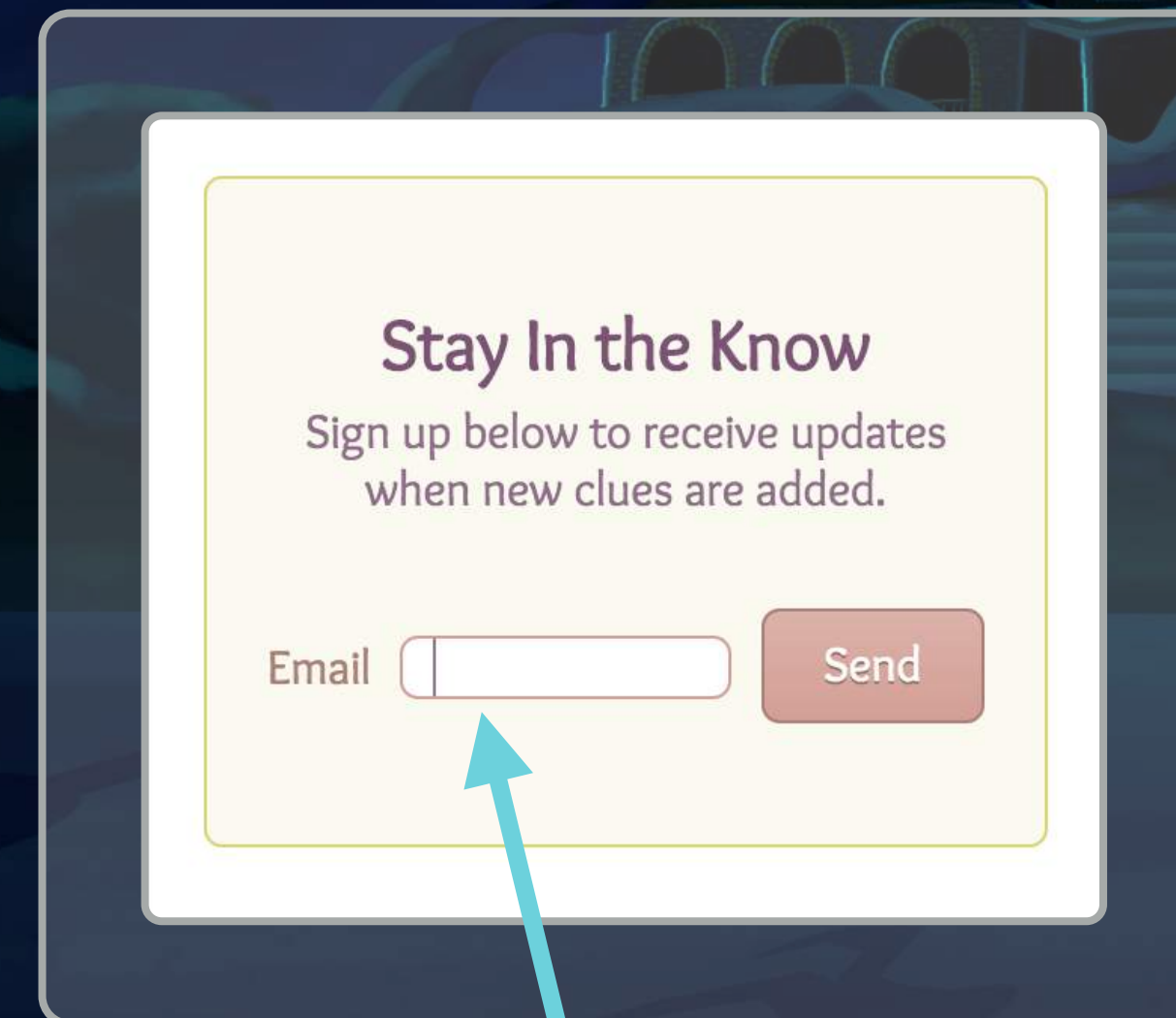
HTML

```
<div ...>
  <label ...>...</label>
  <input class="add-input">
  <button ...>...</button>
</div>
```

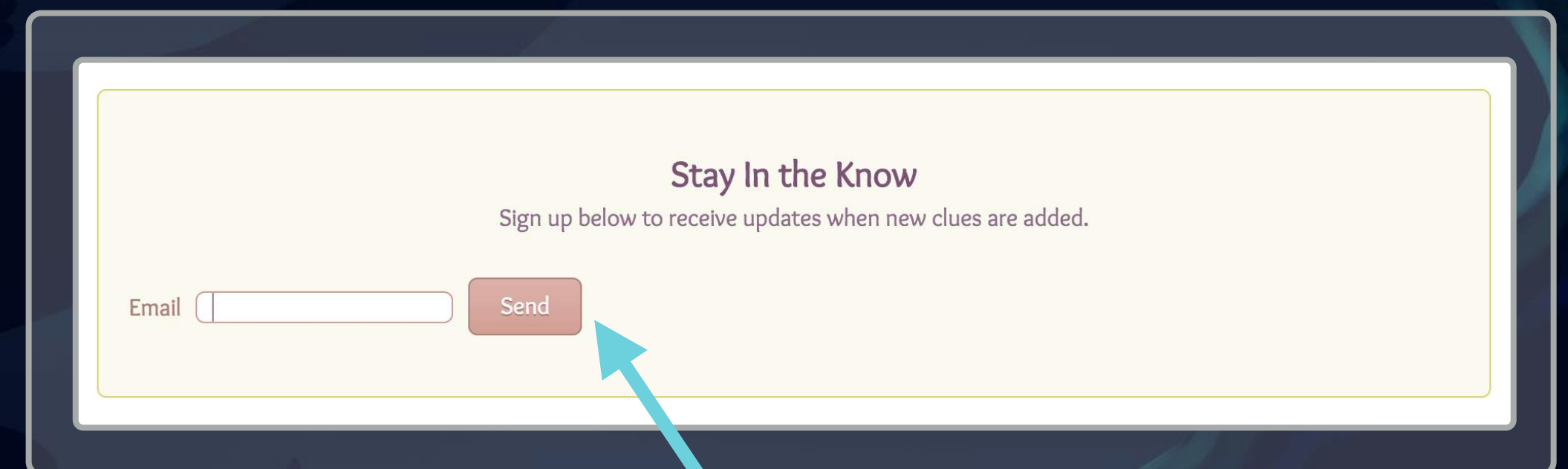
site.css

CSS

```
...
.add-input {
  align-items: center;
  display: flex;
  min-width: 0; }
```



*Set min-width to 0*



*Fills content space*



# flex-grow

Used to specify the ratio of the space an item should fill in the main axis. It accepts numbers and the default is 0.

index.html

HTML

```
<div ...>
  <label class="add-label">...
  <input class="add-input">
  <button class="add-button">...
</div>
```

site.css

CSS

```
...
.add-label,
.add-input,
.add-button {
  flex-grow: 0; }
```

*Items don't grow  
beyond content size*

Stay In the Know

Sign up below to receive updates when new clues are added.

Email



# Using flex-grow for a Single Item

Assigning a value of 1 makes an item fill as much space as possible.

index.html

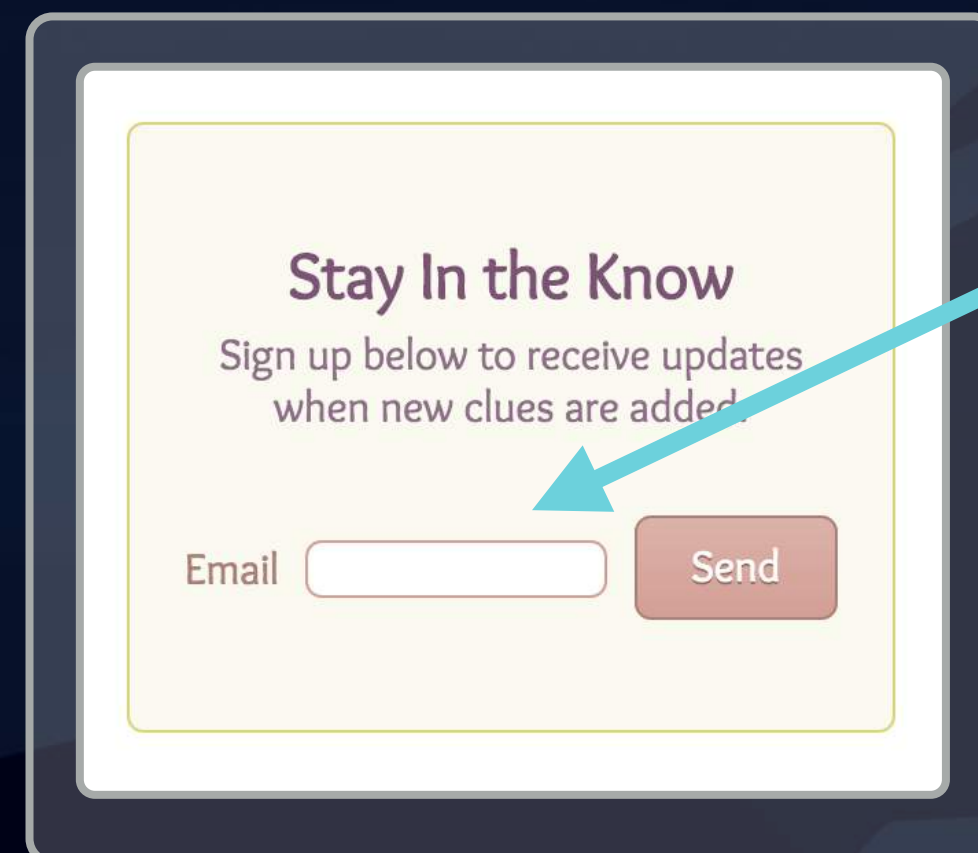
HTML

```
<div ...>
  <label ...>...</label>
  <input class="add-input">
  <button ...>...</button>
</div>
```

site.css

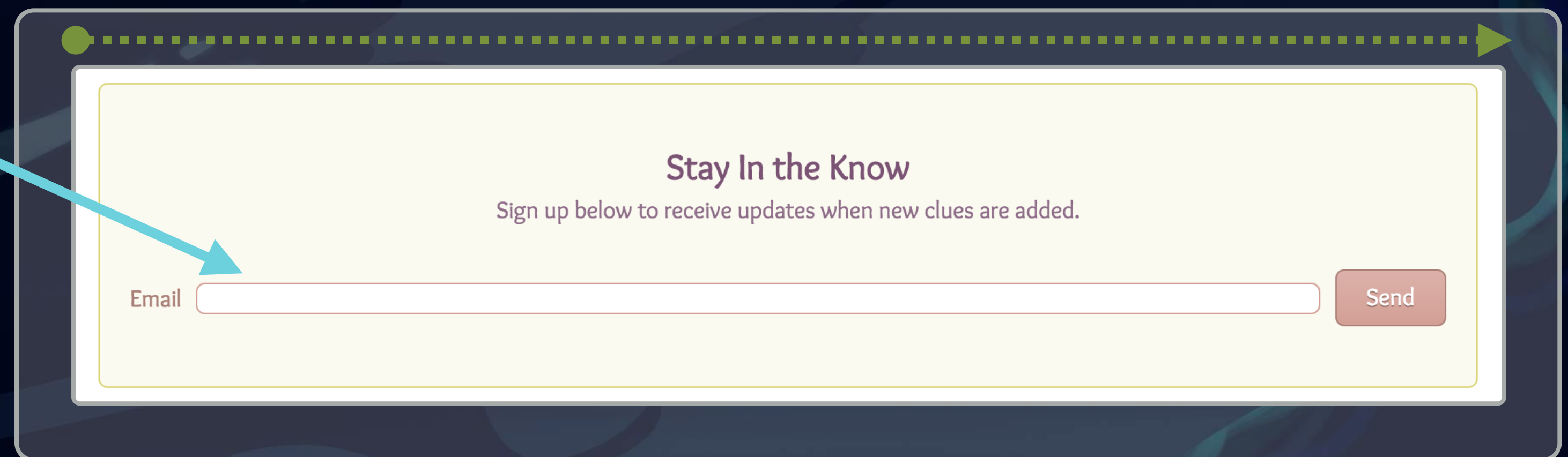
CSS

```
...
.add-input {
  ...
  flex-grow: 1; }
```



A small rectangular form with a yellow border. It contains the text "Stay In the Know" and "Sign up below to receive updates when new clues are added." Below this is a label "Email" followed by a short text input field and a "Send" button.

*Fills all  
available  
space*



A larger rectangular form with a yellow border, representing the same form as the small one but scaled to fill more space. It contains the text "Stay In the Know" and "Sign up below to receive updates when new clues are added." Below this is a label "Email" followed by a long text input field and a "Send" button. A dashed green arrow points from the text "Fills all available space" to the input field.



# Using flex-grow for Multiple Items

Assigning a value of **1** to multiple properties will have them all try to fill as much space as possible.

index.html

HTML

```
<div ...>
  <label class="add-label">...
  <input class="add-input">
  <button class="add-button">...
</div>
```

site.css

CSS

```
...
.add-label,
.add-input,
.add-button {
  flex-grow: 1; }
```

Stay In the Know

Sign up below to receive updates when new clues are added.

Email

*Each item divides  
the space*



# Using flex-grow With Empty Containers

Empty flex containers will divide the space according to the ratio.

index.html

HTML

```
<div class="one">...</div>
<div class="two">...</div>
<div class="three">...</div>
```

site.css

CSS

```
.one,
.three {
  flex-grow: 1; }
.two {
  flex-grow: 2; }
```

1000px

Ratio No Content

.one, 250px

.two, 500px

.three, 250px



# Using flex-grow With Same-sized Content

Content alters how the algorithm divides space.

index.html

HTML

```
<div class="one">Width: 274px</div>
<div class="two">Width: 452px</div>
<div class="three">Width: 274px</div>
```

site.css

CSS

```
.one,
.three {
  flex-grow: 1; }
.two {
  flex-grow: 2; }
```

1000px

Ratio With Content

Width: 274px

Width: 452px

Width: 274px



# Using flex-grow With Uneven Content

Content with different widths will affect the ratio size.

index.html

HTML

```
<div class="one">...</div>
<div class="two">...</div>
<div class="three">...</div>
```

site.css

CSS

```
.one,
.three {
  flex-grow: 1; }
.two {
  flex-grow: 2; }
```

1000px

UnEven Content

One Width: 279px

Two Width: 433px

Three Width: 288px

16 characters

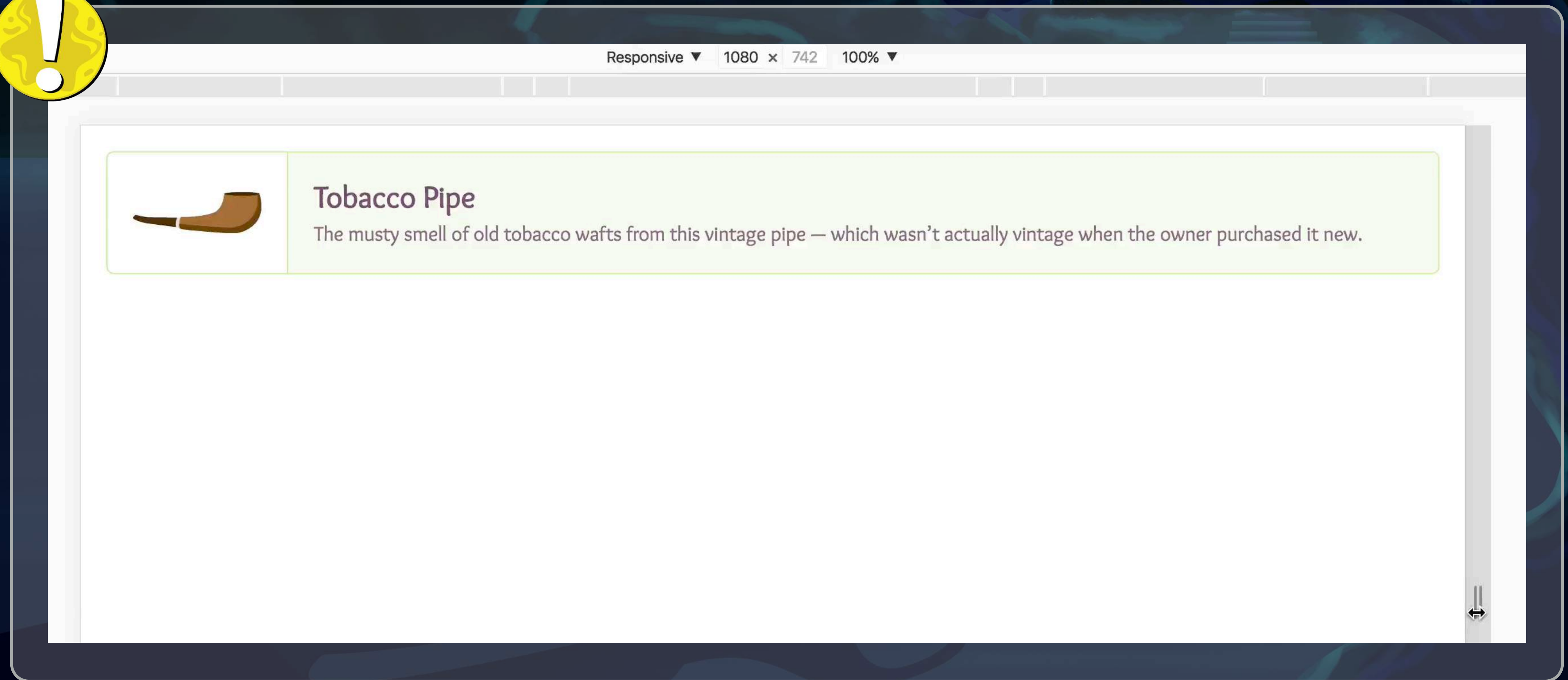
16 characters

18 characters



# Growing Pains

We want the text item to resize, but not the image.





# flex-shrink

Used to specify the "shrink factor" of a flex item. It accepts numbers and the default is 1.

index.html

HTML

```
<article ...>
  <div class="clue-img">...</div>
  <div class="clue-info">...</div>
</article>
```

site.css

CSS

```
...
.clue-img {
  flex-shrink: 0; }
```



## Tobacco Pipe

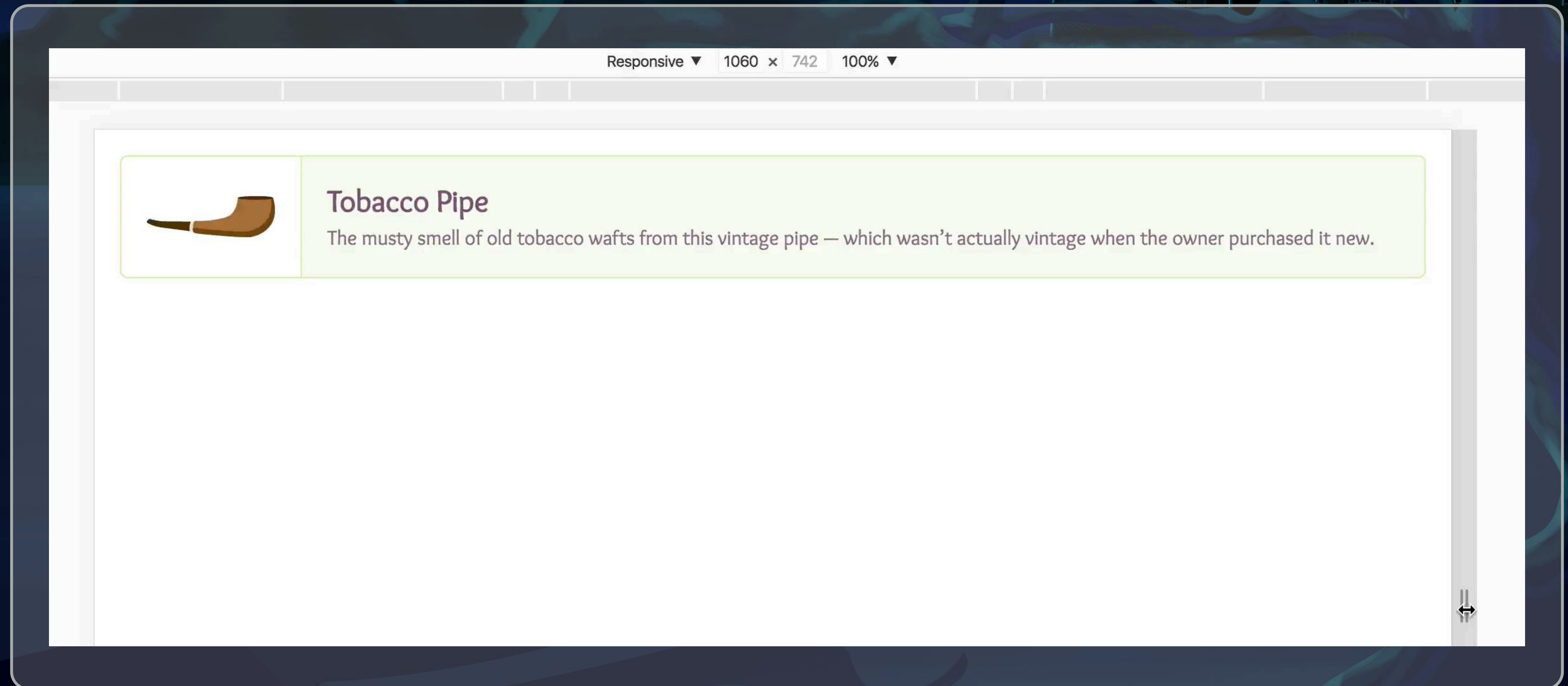
The musty smell of old tobacco wafts from this vintage pipe — which wasn't actually vintage when the owner purchased it new.

*Does not reduce in width*



# flex-shrink: 0 in Action

The `.clue-img` div remains the same size.






CRACKING  
THE CASE WITH

FLEXBOX







Level 4 – Section 2

# Sizing Up the Properties

**New Property for Setting Width Values**



# Problem: Defined Widths for Flex Items



**Brown necktie**

The neutral tones help its owner blend in with any background — especially the faded wallpaper in the aging haunted house.



**Tobacco Pipe**

The musty smell of old tobacco wafts from this vintage pipe — which wasn't actually vintage when the owner purchased it new.



**Gold Pocket Watch**

The telltale ticking of this pocket watch echoes throughout the halls of the haunted house, bouncing off sheet-covered furniture... or ghosts.



**Cane**

There's a dusting of hard-candy sugar along this solid-oak cane, which is ideal for enthusiastic pointing while lecturing whippersnappers.



**Brown necktie**

The neutral tones help its owner blend in with any background — especially the faded wallpaper in the aging haunted house.



**Tobacco Pipe**

The musty smell of old tobacco wafts from this vintage pipe — which wasn't actually vintage when the owner purchased it new.



**Gold Pocket Watch**

The telltale ticking of this pocket watch echoes throughout the halls of the haunted house, bouncing off sheet-covered furniture... or ghosts.



**Cane**

There's a dusting of hard-candy sugar along this solid-oak cane, which is ideal for enthusiastic pointing while lecturing whippersnappers.

*Consistent widths*



# flex-basis

Used to specify the initial size of a flex item. It defaults to `auto` and currently supports CSS units: `%`, `px`, `em`, `rem`, etc.

index.html

HTML

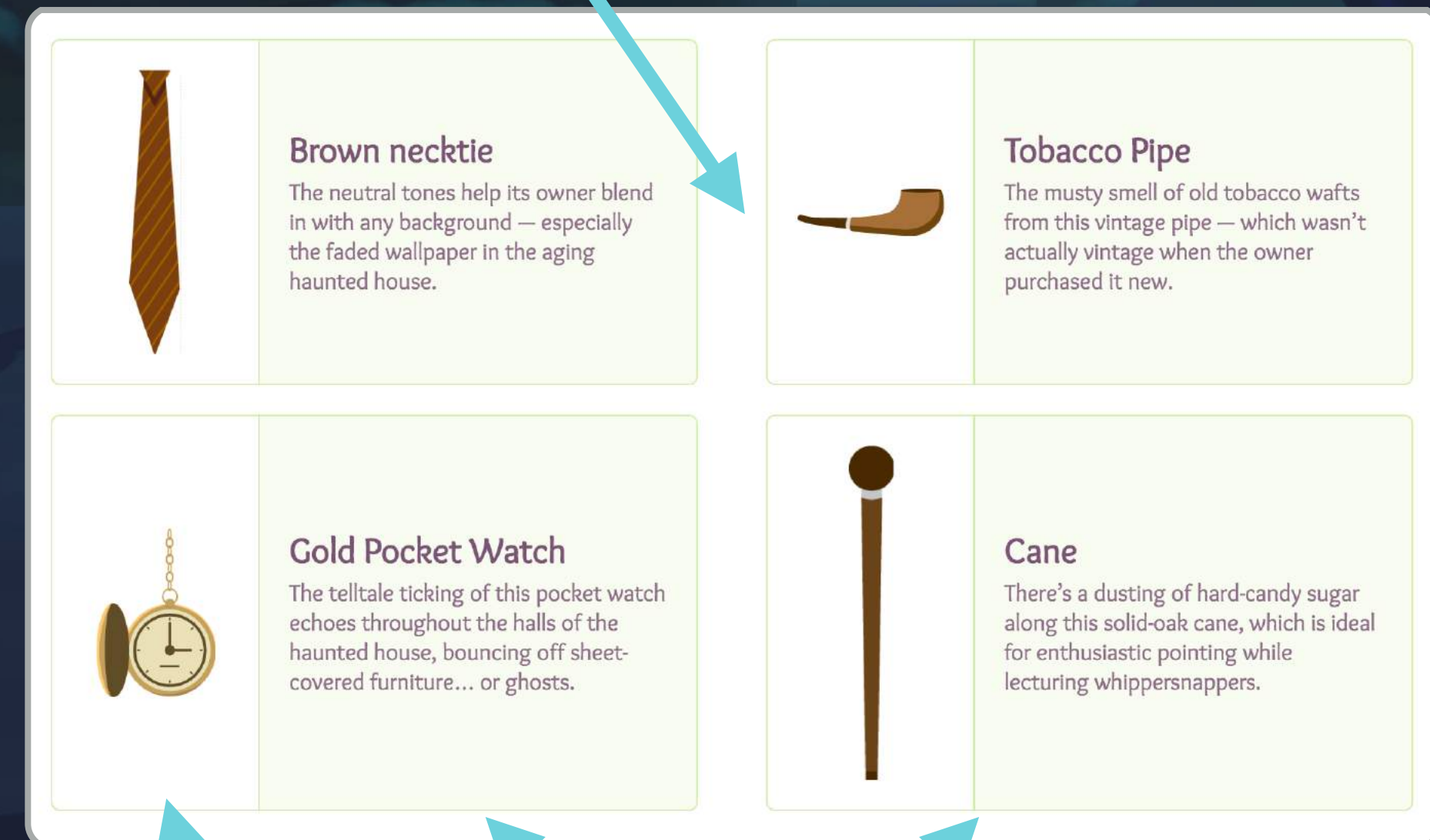
```
<article class="clue">
  <div class="clue-img">...</div>
  <div class="clue-info">...</div>
</article>
```

site.css

CSS

```
...
.clue{
  flex-basis: 47.5%; }
.clue-img {
  flex-basis: 140px; }
```

*justify-content: space-between*



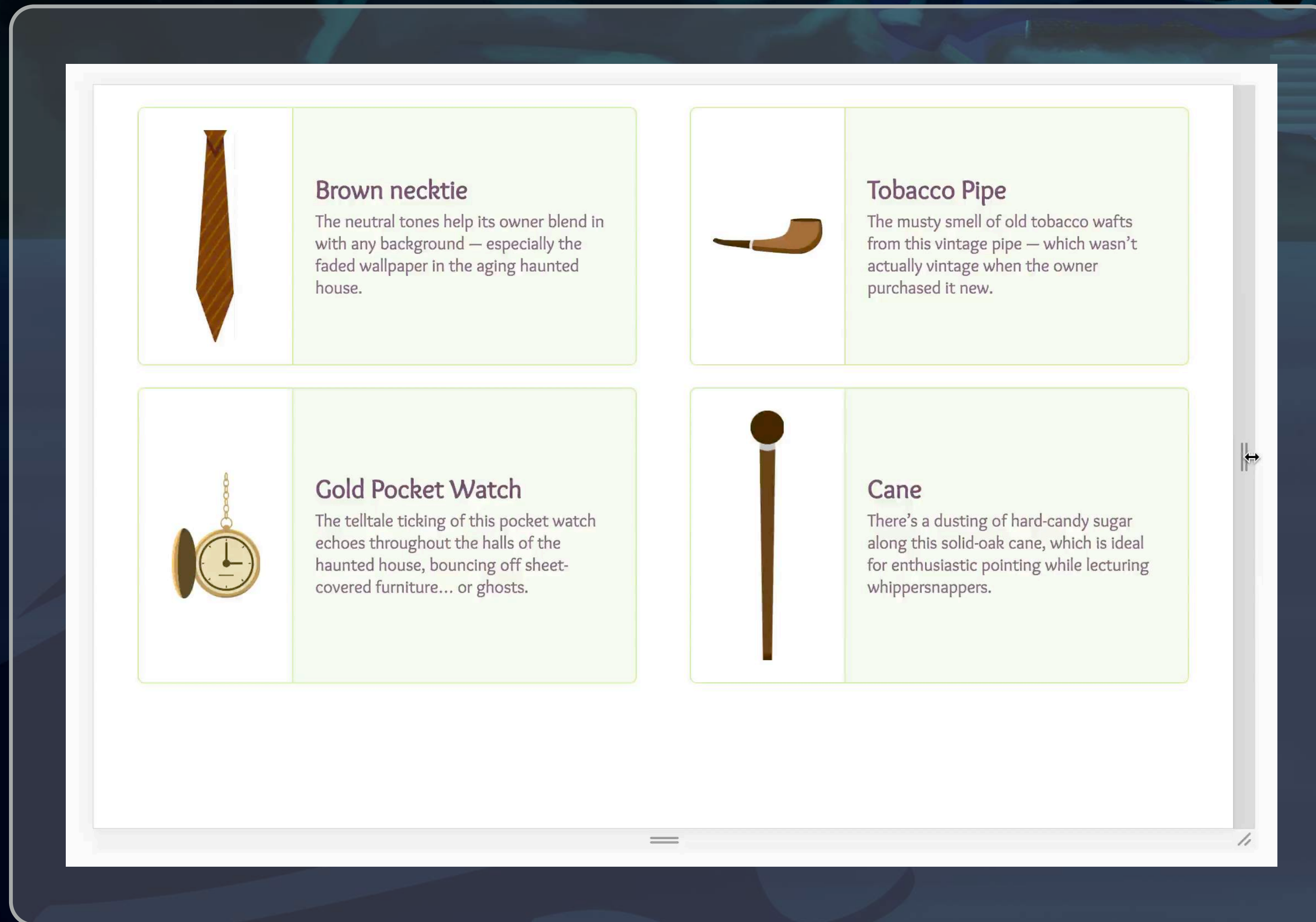
140px

47.5%



# Combining flex-basis and flex-shrink

With one item set to a pixel-based value, the other will adjust to fill the remaining space.





# Using flex-basis and Absolute Units

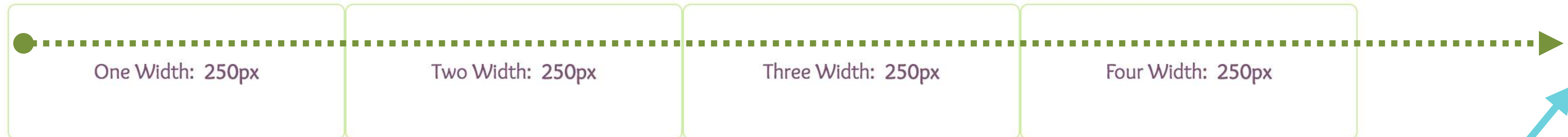
Items with a `flex-basis` that is an absolute unit will not grow beyond the value by default.

site.css

CSS

```
div {  
  flex-basis: 250px; }
```

Pixel



*The flex container is 1166px*



# Using flex-basis and relative units

Items with a relative flex-basis will grow to fill the space.

## Percentage

Width: 290px

Width: 290px

Width: 290px

Width: 290px

## Pixel with flex-grow: 1

One Width: 290px

Two Width: 290px

Three Width: 290px

Four Width: 290px



# More flex-basis Properties — Maybe!

There are more properties that are proposed but not currently implemented in browsers.



```
/* Intrinsic sizing keywords */  
flex-basis: fill;  
flex-basis: max-content;  
flex-basis: min-content;  
flex-basis: fit-content;  
  
/* Automatically size based on the flex item's content */  
flex-basis: content;
```



CRACKING  
THE CASE WITH

FLEXBOX







Level 5 – Section 1

# Property Plotting

Aligning Content in Edge Cases



# Our Layouts Could Use Some Polish

Stay In the Know

Sign up below to receive updates when new clues are added.

Email

Send

*Can we make the heights match?*

*How can we center this?*

## Villainr



Nameson McNamerson

How bad this person is and who they are bad to.



Copyright Honor Amongst Thieves, 2016



# align-self

Used to align individual flex items by overriding the `align-items` value. The default value is `stretch` and it accepts: `stretch` | `flex-start` | `flex-end` | `center` | `baseline`.

index.html

HTML

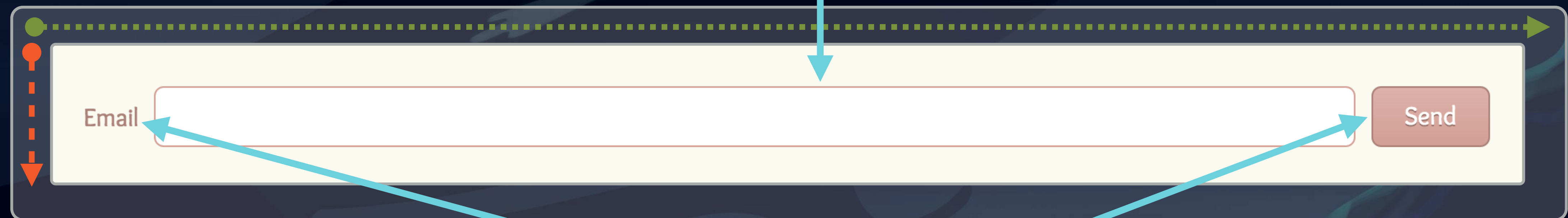
```
<div class="fieldset">
  <label ...>...</label>
  <input class="add-input">
  <button ...>...</button>
</div>
```

site.css

CSS

```
.fieldset{
  align-items: center;
  display: flex; }
.add-input {
  align-self: stretch; }
```

*Stretch on the cross axis*



*Center on the cross axis*



# Using align-self in a Flex Column

index.html

HTML

```
<body ...>
  <header>...<header>
  <main>...</main>
  <footer>...<footer>
</body>
```

site.css

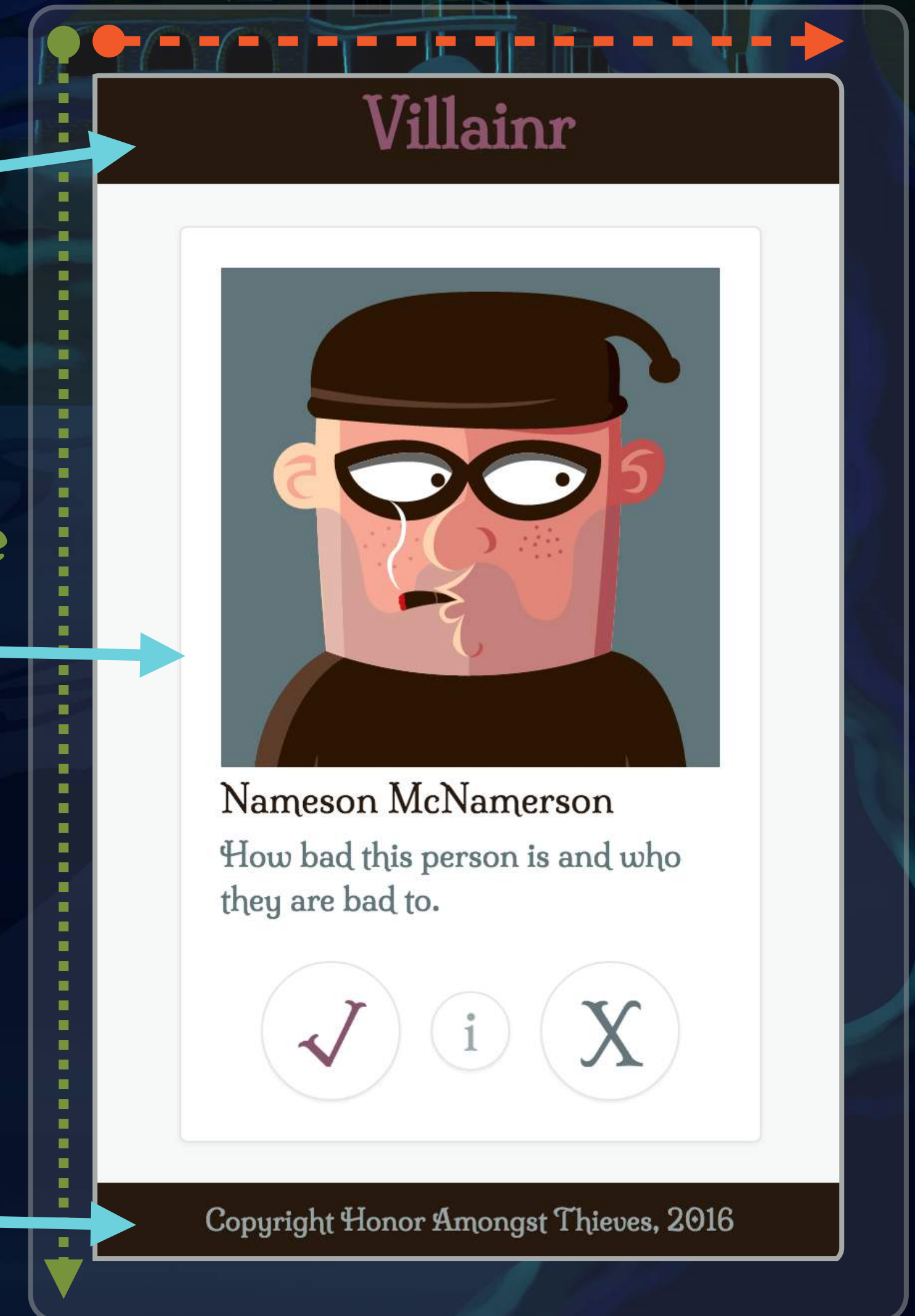
CSS

```
body {
  ...
  align-items: stretch; }
main {
  align-self: center;
  ... }
```

Stretch in the  
cross axis

Center in the  
cross axis

Stretch in the  
cross axis





# Controlling Page Layout With flex-wrap

index.html

HTML

```
<body ...>
  <header>...<header>
  <main>...</main>
  <footer>...<footer>
</body>
```

site.css

CSS

```
body {
  ...
  flex-wrap: wrap;
header, footer {
  ...
  flex-basis: 100%;
}
```

*Each item  
defaults to  
stretch*





# align-items Doesn't Control Wrapped Items

index.html

HTML

```
<body ...>
  <header>...<header>
  <main>...</main>
  <footer>...<footer>
</body>
```

site.css

CSS

```
body {
  ...
  align-items: space-between;
  flex-wrap: wrap; }
header, footer {
  ...
  flex-basis: 100%; }
```

*No change?!*



Nameson McNamerson

How bad this person is and who they are bad to.



Copyright Honor Amongst Thieves, 2016



# align-content

Used to align wrapped flex items. The default value is **stretch** and it accepts: **stretch** | **flex-start** | **flex-end** | **center** | **space-between** | **space-around**.

index.html

HTML

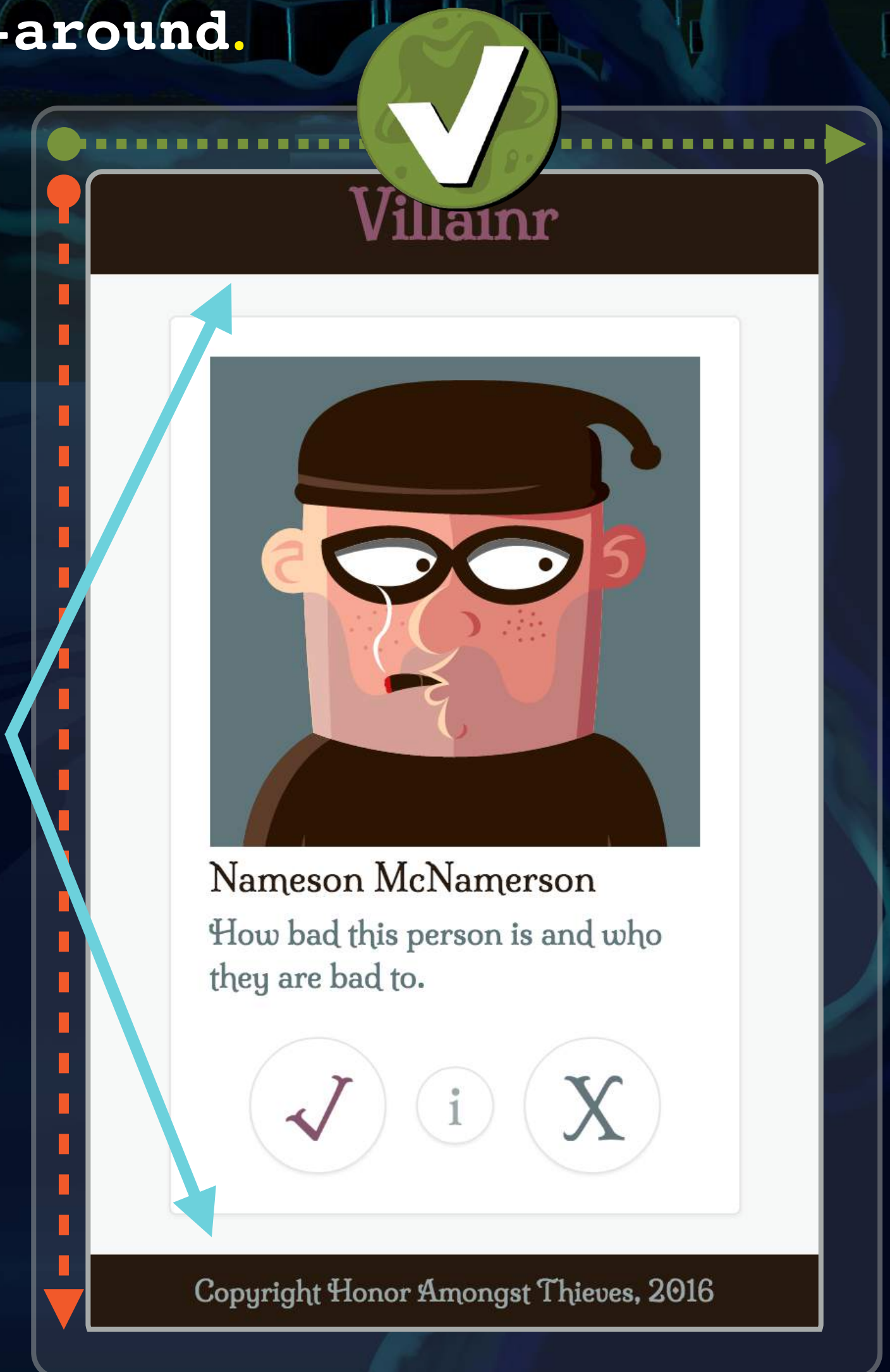
```
<body ...>
...</body>
```

site.css

CSS

```
body {
  ...
  align-content: space-between;
  flex-wrap: wrap; }
header, footer {
  ...
  flex-basis: 100%; }
```

*Space between*






CRACKING  
THE CASE WITH

FLEXBOX







Level 5 – Section 2

# Property Plotting

**Grouping Multiple Properties Into a Single Property**



# flex

Shorthand property used to set values for `flex-grow`, `flex-shrink`, and `flex-basis`. The default is: `0 1 auto`.

site.css

CSS

```
.flex-item {  
  flex-grow: 0;  
  flex-shrink: 1;  
  flex-basis: auto; }
```

site.css

CSS

```
.flex {  
  flex: 0 1 auto; }
```

grow shrink basis

*\*If you have to support older IE, you may want to avoid shorthand.*



# More, Shorter Examples

Single number assigns the grow value

site.css

CSS

```
.flex-item {  
  flex-grow: 1;  
  flex-shrink: 1; /* Default */  
  flex-basis: auto; } /* Default */
```

site.css

CSS

```
.flex {  
  flex: 1; }
```

Two numbers assigns the grow, shrink values

site.css

CSS

```
.flex-item {  
  flex-grow: 1;  
  flex-shrink: 0;  
  flex-basis: auto; } /* Default */
```

site.css

CSS

```
.flex {  
  flex: 1 0; }
```



# More, Short Examples

Single number and percentage assigns the `grow` and `basis` values

site.css

CSS

```
.flex-item {  
  flex-grow: 1;  
  flex-shrink: 1; /* Default */  
  flex-basis: 47.5%; }  

```

site.css

CSS

```
.flex {  
  flex: 1 47.5%; }  

```

None sets `shrink` value, removing flex behavior

site.css

CSS

```
.flex-item {  
  flex-grow: 0; /* Default */  
  flex-shrink: 0;  
  flex-basis: auto; } /* Default */  

```

site.css

CSS

```
.flex {  
  flex: none; }  

```



# flex-flow

Shorthand property used to set values for `flex-direction` and `flex-wrap`. The default is: `row nowrap`.

site.css

CSS

```
.flex-item {  
  flex-direction: row; /* Default */  
  flex-wrap: nowrap; } /* Default */
```

site.css

CSS

```
.flex {  
  flex-flow: row nowrap; }
```

*direction*

*wrap*



# flex-flow Examples

Single direction value sets the **direction** and defaults to **wrap**

site.css

CSS

```
.flex-item {  
  flex-direction: column;  
  flex-wrap: nowrap; } /* Default */
```

site.css

CSS

```
.flex {  
  flex-flow: column; }
```

Single wrap value sets the **wrap** and defaults to **row**

site.css

CSS

```
.flex-item {  
  flex-direction: row; /* Default */  
  flex-wrap: wrap; }
```

site.css

CSS

```
.flex {  
  flex-flow: wrap; }
```



CRACKING  
THE CASE WITH

FLEXBOX

